OPTIONAL FORM NO. 10
5010-104

UNITED STATES GOVERNMENT

Memorandum

Not in

~~X90~~

TO : DCS Programmers

DATE: May 23, 1966

FROM : M. K. Chartz

SUBJECT: New Operating Procedures for DCS

1. Effective May 9, 1966 the DCS began operating 24 hours per day, five days a week except for the time required by IBM to maintain the equipment. This action was necessary in order to meet the growing workload and to improve turn-around time. However, to achieve these ends the following operating procedures must be established.

   a. Check-out jobs ≤ 5 minutes will be run as they are submitted on prime shift.

   b. All production jobs ≤ 2 minutes will be run as they are submitted on prime shift.

   c. Check-out jobs $> 5 \leq 10$ minutes will be run as submitted under the following conditions:

      1) On prime shift one of these jobs will be run every two hours.

      2) All jobs in the shop at 4:30 p.m. should be completed by 8:00 a.m. the next morning.

   It is hoped that this schedule will give jobs of this length 3 shots every 2 days.

2. After the existing backlog has been removed it is hoped that all jobs ≤ 15 minutes will have a 24 hour turn-around time and that jobs of an hour or more will be returned to the user within a week. It may be a few weeks before this goal can be achieved.

3. In this proposed environment the remote stations will have no special priorities. However, they will observe the following restrictions:

a. The restrictions in 1a. to 1c. also apply to the remotes.

b. While there is no limit to the number of jobs a station may submit, no more than two tapes may be in SETUP at one time.

c. It is recommended that jobs > 5 minutes be brought to CAB at 4:30 p.m. They should then be completed by 8:00 a.m. the next day. They may be submitted from the remote with the risk of being lost if there is a system failure.

d. Each branch may establish its own operating procedures; however, they must be more restrictive than those stated in this memorandum.

e. As of May 16, 1966, there is an operator at the 7740 console. Questions may be transmitted to the console typewriter to minimize the telephone calls. Also notice of system changes will be sent out on the typewriter when these are implemented.

4. While CAB welcomes suggestions for change, we would like to try these rules for a two month period and then discuss revisions. During this period we will not distribute red priority cards. The scheduling of any critical project will be handled on an individual basis by myself or someone I have designated.

5. In order to expedite service calls on all equipment, the users are requested to call Nancy Wright, extension 2208 and report the difficulty. She will contact the customer engineer. For weekend or night service, the user may call 248-2580.

Marcelline K. Chartz

Marcelline K. Chartz
Staff Specialist

MKC:nw

OPTIONAL FORM NO. 10
5010-101

UNITED STATES GOVERNMENT

Memorandum

TO : DCS Programmers

DATE: April 21, 1966

FROM : Ben Meyer

SUBJECT: Use of Tapes

1. In order to improve operations of the DCS the DCS operator needs to know the total number of tapes used by a job, so that the job can be scheduled correctly. It is therefore necessary that all tapes used by a job be listed on the EØF card when the job is submitted. This includes all scratch tapes and print tapes.

2. The programmer should make a note of the above in sections: 6.2.3.2, 7.2.3.2, and 8.4.5 of his DCS USER'S MANUAL.

3. The card shown below is an example of how to fill out an EØF card.

ENC NO.	FOR	DATE	EST. TIME	PHONE	MAIL
PF2127	John Doe	4/21/66	3.00	2208	

CONTROL CARDS

JOB	$ID
	$IBJOB
	$DATA
DECK	$IBFTC
	$IBMAP

BINARY DECK MAKEUP

LOGICAL TAPE NO.	MOUNT	SAVE	FILE PROTECT	PRINT	PUNCH
7	PLOT				
8	1545	✓			
9	PRINT				
10	SCRATCH				
12	SCRATCH				

IBM L58838

OPTIONAL FORM NO. 10
5010-104

UNITED STATES GOVERNMENT

Memorandum

TO : DCS Programmers DATE: December 23, 1965

FROM : M. K. Chartz

SUBJECT: Update for Direct Couple System Users' Manual

1. Enclosed is an update for the Direct Couple System Users' Manual. In addition to corrections and clarifications, this update contains operating information to enable the programmers to use binary decks from disk. Starting January 10, 1966 the list of binary decks on the EOF card will be replaced by the appropriate $DECK cards. It is recommended that the users read the pertinent sections of the manual so as to familiarize themselves with the new deck maintenance procedures. If there are any questions, a member of CAB will be glad to assist you.

2. Included in this update are the deck makeup diagrams. However, these do not include the $DECK cards.

3. Also, FORTRAN II should not have the two *-cards when Debug Macro cards are used.

M. A. Merriman for

Marcelline K. Chartz
Staff Specialist

MKC:fg

Enclosure:
As stated above

THE AMES 7040/7094 DCS USER'S MANUAL (1965)

Nasa Ames

JULY 1965

COMPUTATION & ANALYSIS
BRANCH

THE AMES 7040/7094 DCS USER:S MANUAL

→ 1.0.0	Nov/65	TABLE OF CONTENTS
→	Effective Date:	Nov 2, 1965
2.0.0	Jul/65	INTRODUCTION
3.0.0	Jul/65	UPDATING PROCEDURE
4.0.0	Jul/65	EQUIPMENT
4.1.0	Jul/65	7040/7094 Direct Coupled System
→ 4.2.0	Nov/65	Remote Stations
4.3.0	Jul/65	Plotters
4.4.0	Jul/65	EAM
4.5.0	Jul/65	Special Equipment
4.6.0	Jul/65	1401
5.0.0	Jul/65	7040/7094 DCS OPERATING SYSTEM
5.1.0	Jul/65	Introduction
5.2.0	Jul/65	DCMUP
5.3.0	Jul/65	IBSYS
5.3.1	Jul/65	Description
5.3.2	Jul/65	Block Diagram
5.4.0	Jul/65	Input/Output Control System (IOCS)
5.5.0	Jul/65	Operating System Definitions
→ 5.6.0	Nov/65	Binary Deck Maintenance
→ 5.6.1	Nov/65	Introduction
→ 5.6.2	Nov/65	Use of Decks on Disk
→ 5.6.3	Nov/65	Binary Decks at Compile or Assembly Time
→ 5.6.4	Nov/65	Purging Binary Decks From Disk
→ 5.6.5	Nov/65	Adding Binary Decks to Disk
→ 5.6.6	Nov/65	List of Binary Decks on Disk
→ 5.6.7	Nov/65	Proudction Decks and Disk
→ 5.6.8	Nov/65	Card Copies of Binary Decks
→ 5.7.0	Nov/65	Operating System Restrictions
→ 5.7.1	Nov/65	Use of Sense Switches on the 7094
→ 5.7.2	Nov/65	Use of PAUSE, HLT, HPR on the 7094
→ 5.7.3	Nov/65	Mounting of Tapes
→ 5.7.4	Nov/65	FORTRAN II Chain Jobs
6.0.0	Jul/65	IBJOB MONITOR SYSTEM
6.1.0	Jul/65	Function
6.2.0	Jul/65	FORTRAN IV
→ 6.2.1	Nov/65	Job-Makeup
6.2.2	Jul/65	Sample FORTRAN IV Job
6.2.3	Jul/65	Control Cards
6.2.3.1	Jul/65	PLOT
→ 6.2.3.2	Nov/65	EOF
6.2.3.3	Jul/65	$JOB
6.2.3.4	Jul/65	$SETUP
→ 6.2.3.5	Nov/65	$IBJOB

	6.2.3.6	Jul/65	$IBDBL
	6.2.3.7	Jul/65	*DEBUG
	6.2.3.8	Jul/65	*ETC
	6.2.3.9	Jul/65	*DEND
	6.2.3.10	Jul/65	$IBFTC
→	6.2.3.11	Nov/65	$DECK
→	6.2.3.12	Nov/65	$DATA
→	6.2.3.13	Nov/65	ADDITIONAL CARDS
	6.3.0	Jul/65	Debugging
	6.3.1	Jul/65	Introduction
	6.3.2	Jul/65	Compile-Time Requirement
	6.3.3	Jul/65	Load Time Requirement
	6.3.4	Jul/65	Debug Request Cards
	6.3.5	Jul/65	Debug Request Statements
	6.3.5.1	Jul/65	Table of Debug Req.Statements
	6.3.5.2	Jul/65	Brief Description of Debug Request Statements
	6.3.6	Jul/65	Debug Request and FORTRAN Statement Differences
	6.3.7	Jul/65	Notes on the Debugging Package
	6.3.8	Jul/65	Examples
→	6.3.8.1	Nov/65	Post Mortem Dump Example
	6.3.8.2	Jul/65	Dynamic Dump Example
	6.3.9	Jul/65	System Debugging Aids
	6.3.9.1	Jul/65	Error Walk-Back Feature
	6.3.9.2	Jul/65	Floating Point Overflow
→	6.3.9.3	Nov/65	Input/Output Error Messages Generated by the 7040
	6.4.0	Jul/65	Job Termination
	6.5.0	Jul/65	Description of IBJOB Output
→	6.6.0	Nov/65	Overlay (Storage Allocation for Large Jobs)
→	6.6.1	Nov/65	Purpose
→	6.6.2	Nov/65	Description
→	6.6.3	Nov/65	Deck Makeup
→	6.6.4	Nov/65	Note
	6.7.0	Jul/65	MAP (Macro Assembly Program)
	6.8.0	Jul/65	COBOL (COmmon Business Oriented Language)
	7.0.0	Jul/65	FORTRAN MONITOR SYSTEM (FMS)
	7.1.0	Jul/65	Function
	7.2.0	Jul/65	FORTRAN II
→	7.2.1	Nov/65	Job Makeup
→	7.2.2	Nov/65	Sample FORTRAN II Job
	7.2.3	Jul/65	Control Cards
	7.2.3.1	Jul/65	PLOT
→	7.2.3.2	Nov/65	EOF
	7.2.3.3	Jul/65	$JOB
	7.2.3.4	Jul/65	$SETUP
	7.2.3.5	Jul/65	$EXECUTE FORTRAN
	7.2.3.6	Jul/65	XEQ
	7.2.3.7	Jul/65	LABEL
	7.2.3.8	Jul/65	SYMBOL TABLE
	7.2.3.9	Jul/65	DEBUG
→	7.2.3.10	Nov/65	Additional * Cards

→ 7.2.3.11	Nov/65	$DECK
→ 7.2.3.12	Nov/65	DATA
7.3.0	Jul/65	Source Language Debugging at Object Time
7.3.1	Jul/65	Introduction
7.3.2	Jul/65	Compilation
7.3.3	Jul/65	Execution
7.3.4	Jul/65	Debug Macro Cards
7.3.4.1	Jul/65	Name Card
7.3.4.2	Jul/65	Dump Cards
7.3.5	Jul/65	Notes
7.3.5.1	Jul/65	Array Restrictions
7.3.5.2	Jul/65	G-Format
7.3.5.3	Jul/65	Timing
7.3.5.4	Jul/65	Placement of Dump Cards
7.3.5.5	Jul/65	Storage Requirements at Execute Time
7.3.5.6	Jul/65	Table Limits
7.3.5.7	Jul/65	Output Limit
7.3.5.8	Jul/65	CONTINUE Statements
7.3.5.9	Jul/65	Use of Names in Debug Macros
7.3.6	Jul/65	Sample Program
7.4.0	Jul/65	Post Mortem Dumping
7.4.1	Jul/65	Introduction
7.4.2	Jul/65	CRISIS
7.4.3	Jul/65	POST
7.5.0	Jul/65	Conversion of FORTRAN II to FORTRAN IV
7.6.0	Jul/65	FAP (FORTRAN Assembly Program)
8.0.0	Jul/65	GENERAL DCS OPERATING INFORMATION
8.1.0	Jul/65	Job Priorities
8.2.0	Jul/65	Job Handling
8.2.1	Jul/65	Use of EAM Equipment
8.2.2	Jul/65	Keypunching Standards
→ 8.2.3	Nov/65	Job Submittal
8.2.4	Jul/65	Job Pick-up
8.3.0	Jul/65	Job Accounting
8.3.1	Jul/65	CAB Number Assignment
8.3.2	Jul/65	Machine Rental Charges
8.3.3	Jul/65	Work Orders
8.3.4	Jul/65	CAB Accounting Procedure
8.4.0	Jul/65	Magnetic Tape Usage
8.4.1	Jul/65	Introduction
8.4.2	Jul/65	Definitions
→ 8.4.3	Nov/65	Logical Tape and Buffers
8.4.4	Jul/65	DCS Tape Information
8.4.4.1	Jul/65	Introduction
8.4.4.2	Jul/65	Density
8.4.4.3	Jul/65	Record Size
8.4.4.4	Jul/65	Deblocking of Output Tapes
8.4.4.5	Jul/65	Blocking of Input Tapes
→ 8.4.4.6	Nov/65	Maximum Number of Files

8.4.5	Nov/65	End-of-File Card
8.4.6	Jul/65	Magnetic Tape Library
8.4.7	Jul/65	Recommended Tape Usage Procedures
8.4.8	Jul/65	Subroutines
8.5.0	Jul/65	Plotter Usage
8.5.1	Jul/65	Introduction
8.5.2	Jul/65	Plotting Hardware
8.5.3	Jul/65	Description of Subroutines
8.5.3.1	Jul/65	Plot Page
8.5.3.2	Jul/65	Subroutines
8.5.4	Jul/65	Preparation of Plots
8.5.4.1	Jul/65	Control Cards
8.5.4.2	Jul/65	Maximum Point Displacement Control
8.5.4.3	Nov/65	Restrictions
8.6.0	Jul/65	Subroutine Abstract Usage
9.0.0	Jul/65	SPECIALIZED INPUT/OUTPUT INFORMATION
9.1.0	Jul/65	Paper Tape Usage
9.1.1	Jul/65	Experimental Design
9.1.2	Jul/65	Paper Tape Standards
9.1.3	Jul/65	Dictionary
9.1.4	Jul/65	Proposed American Standard - One-inch Performated Tape for Information Interchange
9.1.5	Jul/65	Miscellaneous Notes
9.1.6	Jul/65	Program for Punching Paper Tape
9.1.7	Jul/65	Program for Reading Paper Tape
9.2.0	Jul/65	Analog to Digital Conversion of Magnetic Tapes

Appendix A

1.0.0	Nov/65	DIRECTORY
1.1.0	Nov/65	Physical Plant
1.2.0	Nov/65	Personnel

Appendix B

1.0.0	Jul/65	CONVERSION OF A FORTRAN II PROGRAM TO A FORTRAN IV PROGRAM
1.1.0	Jul/65	Introduction
2.0.0	Nov/65	USAGE OF SIFT
2.1.0	Jul/65	Discussion
2.2.0	Nov/65	Restrictions

Index

2.0.0 Jul/65 INTRODUCTION

The purpose of this manual is (1) to give the user a brief description of the Ames IBM 7040/7094 Direct Couple System and its associated peripheral equipment and (2) to give the user the information necessary for processing a computer program on the above-mentioned system.

3.0.0 Jul/65 UPDATING PROCEDURE

The Table of Contents serves as a guide for the use of this manual. It will also be used as an aid in keeping this manual current. To the left of each subject title, are given a section number and a date. The section number specifies the location of the section in the manual, and the date specifies the issue date of the section. The corresponding sections of the text are also dated, and a current manual exists when the dates in the text agree identically with the dates in the Table of Contents. A Table of Contents is current if its issue date agrees with that of the master copy in the office of the CAB secretary. Sections flagged by an arrow in the Table of Contents have been modified in that update. In the text, the exact changes are also noted by an arrow in the margin. To illustrate, if an entry in the Table of Contents reads:

6.2.3 <u>Control Cards</u>
Nov/65

and the section in the manual reads:

6.2.3 <u>Control Cards</u>
Jul/65

the latter is not the current version. It should be replaced by the current version which would read:

6.2.3 <u>Control Cards</u>
Nov/65

4.0.0 Jul/65 EQUIPMENT

Available to the research scientists who are programming are the IBM Direct Couple System (DCS), consisting of a 7040 and 7094, two Electronic Associates Magnetic Tape Plotters, and various Electronic Accounting Machines (EAM). In addition, the Direct Couple System has facilities for handling four remote stations.

4.1.0 Jul/65 <u>7040/7094 Direct Coupled System</u>

This system involves two separate and distinct computers, the 7040 and the 7094, whose memories are linked by a direct connection. In essence, the 7094 has the standard 32,768 words of core memory but no input/output channels and therefore has no means of receiving or transmitting information except through the core of the 7040. The 7040 has the same amount of memory, 32,768 words, but in addition it has four input/output channels to which are connected the on-line 1401, an 800 word per minute reader,

4.1.0 Jul/65 (continued)

a 250 card per minute punch, two 600 line per minute printers, 11 magnetic tapes, two 1301 disks for bulk storage, and a 7740 Communication Control System. All these devices are controlled by the 7040, and the information which the 7094 needs for processing is provided as required. Such a machine configuration enables the 7094 to operate as a compute processor while the 7040 assumes the responsibility of scheduling all input/output operations. The connection which links the memories of the two computers transfers information at the rate of 375,000 characters per second. The 1301 disks are two module disks each capable of storing 56,000,000 characters. All the magnetic tapes operate at 112.5 inches per second and can handle densities of 200, 556, and 800 characters per inch. Thus the maximum transfer rate is 90,000 characters per second.

4.2.0 Nov/65 <u>Remote Stations</u>

In an attempt to improve the communication between the research scientist and the central computer, a 7740 Communications Control unit has been installed. By the use of telephone connections and Data-phones, information can be transmitted to and from the 7740 to locations removed from the central area. At these remote locations are IBM 1440 computers which essentially provide input/output capability. These units each have 4,000 characters of memory, a 400 card per minute reader, a 240 line per minute printer, and a console inquiry station with a Selectric typewriter for communication with operation personnel. Such units will accept a job at a remote location and transmit it via the telephone lines to the 7740 where it will be entered into the input queue of the 7040 with the appropriate priority. It then is handled as any other job in the system and its output is placed on disk. However the output is also transmitted over the telephone lines to remote where it is printed. Presently there are four IBM 1440 computers installed at the following locations:

1)	Building N-207	Mission Analysis Division
2)	Building N-210	Theoretical Guidance & Control Branch
3)	Building N-230	Physics Branch
4)	Building N-223	Theoretical Studies Branch

These remote computers can be used by any programmer provided arrangements are made with the particular branch where the computer is located. Operational details are maintained at each location.

4.3.0 Jul/65 <u>Plotters</u>

Also available to the staff are two EAI Dataplotters (Model 3440), which accept only magnetic tape as input data. Each is equipped with a pen and a 48 character symbol printer. These plotters have the ability to plot points or lines in several colors with the pen or print any of 48 alphanumeric or special characters with the printer. They can also "freeline" plot, i.e., plot with fairing, at speeds up to 4,500 points per minute. Plots may be reorigined and the scaling changed by information on the magnetic tapes.

For complete details concerning the use of these plotters, see section 8.5.0 on Plotter Usage.

4.4.0 Jul/65

EAM

In the EAM department there are keypunches, reproducers, a sorter, a collator, and listers, which are used by the personnel in the EAM group to meet the needs of the programming staff.

4.5.0 Jul/65

Special Equipment

There are still two special on-line devices on the 7040/7094 Direct-Coupled System (DCS). The first of these is an on-line 1401 which is for the exclusive use of the Theoretical Branch. This machine has a 4,000 character memory, a card reader and punch, a printer, and three dual density tape drives. In addition, it has an on-line Electronics Associates Plotter. This 1401 appears as a priority reader-punch-printer station to the 7040 and upon demand interrupts the DCS to process the job which is entered through the 1401 station. The second on-line device is a connection to the analog computer located at the AFS Branch. This connection is made by means of a Packard Bell linkage system which contains analog to digital and digital to analog converters. When this system is in operation it permits the two computers to operate together as a single hybrid system. This system has application currently in the guidance and control area but could be applied to other problems which have the demands of fast response and high accuracy. When in use it has the highest priority and interrupts immediately any operating job on the DCS.

4.6.0 Jul/65

1401

The 1401 has 12,000 characters of memory and four magnetic tapes and is used primarily for administrative data processing.

5.0.0 Jul/65 7040/7094 DCS OPERATING SYSTEM

5.1.0 Jul/65 Introduction

A fundamental concept of Direct-Couple System (DCS) is that many jobs are handled by the system simultaneously. To facilitate control over the many jobs being handled concurrently, each job is divided into three phases: preprocessing, processing, and postprocessing. Control of the system is exercised by two supervisory programs: the DCS Multiprocessor (DCMUP) and the IBM 7090/7094 IBSYS monitor. DCMUP resides in the 7040 and exercises control over both the preprocessing and post-processing phases of a job. IBSYS resides in the 7094 and exercises control over the processing phase of a job. DCMUP and IBSYS perform their functions asynchronously.

5.2.0 Jul/65 DCMUP

The most significant functions of DCMUP are that it handles all input/output for the 7094 and maintains the workload for the 7094. It schedules jobs based on their individual priorities and queues them for 7094 processing. DCMUP enters jobs from the card reader directly into an input queue on the disk. As the 7094 finishes a job it requests a new job from the 7040. At the time of this request, the 7040 selects the top job from the input queue and transmits it to the 7094. When the 7094 is processing that job it receives all information from the 7040 which will have stored it on an input/output device. All systems including FORTRAN II and FORTRAN IV are in residence on the disk and are brought in as required. As output is generated by the object program it is transmitted through the 7040 to the disk. When the job is completed on the 7094, it is entered into the output queue. This queue (the output queue) is serviced by the printers operating on the 7040. As soon as a printer is free, the highest job in the output queue starts to print. Priorities in the input queue are determined by time and line count estimates (to be explained later). The output queue is processed in the order in which the jobs were computed. If, however, it is necessary to expedite a job, this can be done by loading the job with an overriding priority.

The functions of the 7094 operator in controlling job processing are performed by DCMUP. The operator communicates with the system through the 7040 console keys or through a 1014 Inquiry Unit. Facilities are included for job status inquiry, change of job priority, system restart, and setup communication.

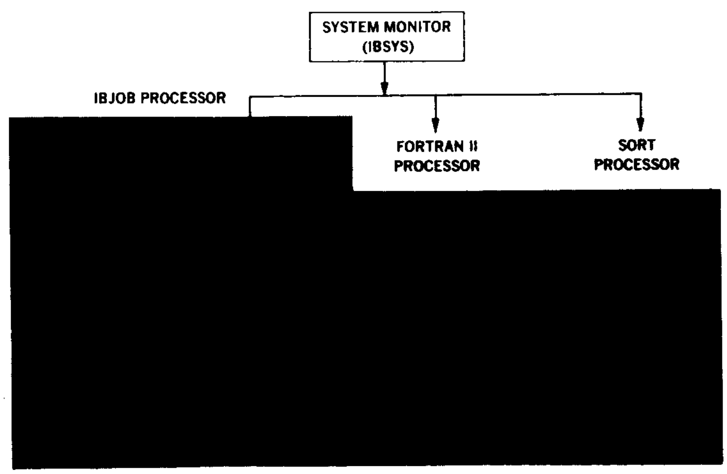
5.3.0 Jul/65 <u>IBSYS</u>

5.3.1 Jul/65 <u>Description</u>

As mentioned earlier the IBSYS monitor resides in the core of the 7094, and its principal role is to exercise control over job execution. It does not initiate input/output activity but merely requests the activity which is then performed by DCMUP. The original concept of a monitor is a program which accepts as input a variety of jobs and processes each job as a separate entity. For example, a job could be composed of many segments, or decks, and these segments could have different physical forms. They could be FORTRAN decks, MAP or FAP decks, or binary decks. The monitor processes each deck as required and then loads the necessary object decks for the job execution phase. The IBSYS monitor, however, is of higher level than the job monitor just described. It is essentially a monitor's monitor in that it controls the monitor which is going to control the job. The advantage of this method of operation is that there can be continuous processing which may involve a mixture of jobs as well as systems (job monitors). Under IBSYS it is possible to process a group of FORTRAN II jobs, then switch to FORTRAN IV for some jobs, and then do sorting jobs. In this case the IBSYS monitor would call the FORTRAN monitor system to do the FORTRAN II jobs; it would call the IBJOB monitor to do FORTRAN IV jobs; and then it would call the IBSORT monitor to do the sorting jobs. As is obvious, a system of this type permits greater flexibility of operation and allows a greater variety of jobs to be processed without operational complexities. The following is a block diagram of BASIC IBSYS on DCS.

5.3.2 Jul/65

THE IBSYS OPERATING SYSTEM



5.4.0 Jul/65

Input/Output Control System (IOCS)

Most input/output for jobs operating under the IBJOB monitor is controlled by the Input/Output Control System (IOCS). This system schedules input/output so as to maximize the usage of input/output devices and to minimize the time the computing process is delayed waiting for 7040 recognition of input/output demands. To achieve this result, IOCS makes use of that part of core storage still available after a job has been loaded. This available storage is divided into many sections, each one of which can hold either one line of print, one card of input, or one binary record. These sections are called buffers and are used to hold I/O information for the program. The actual servicing of the I/O devices is done from these buffers. Thus, a READ or a WRITE statement transfers information between the buffer area and the program storage area, and IOCS fills or empties the buffer areas as the needs are made known. This means that during job execution, data are read into the buffer area before the READ statement for those data is executed, and the READ statement merely places them in the correct program locations. Similarly the WRITE statement moves data from program storage into a buffer area, and at a later time IOCS writes the information on tape. While the placement of the READ and WRITE statements is important in maintaining some type of I/O continuity, the programmer does not have to be concerned with the timing of the IOCS operation. His data will always be in the correct locations at the time he wants to use them. If a job is primarily a compute job, then the number of buffers which are available is not too critical because the buffers will be emptied before the program is ready to fill them again. On the other hand, if it has repeated bursts of I/O, then a significant number of buffers would be an asset in minimizing the amount of time the computer waits for I/O servicing. For this reason, in large programs the programmer should consider the size of his arrays and the way he plans his I/O in order to achieve a moderately efficient program.

The above is a very brief description of the functional aspects of IOCS, but the programs which actually perform these operations are part of a highly complex and carefully integrated system. Thus, errors in input/output can produce rather complicated interactions within this particular system and at times can even affect the other systems.

5.5.0 Jul/65

Operating System Definitions

The following sections give the programmer the information necessary for processing jobs which run under the IBJOB monitor (FORTRAN IV, MAP, and COBOL jobs) or under the FORTRAN monitor (FORTRAN II and FAP jobs). To simplify these discussions a few definitions are given:

(continued)

(1) Deck - A set of cards in either source language form or binary form for a single program, e.g., a FORTRAN main program or subroutine.

(2) Job - A set of decks to be processed on the computer.

(3) Run - A continuous flow of jobs to be processed on the computer.

(4) Standard Monitor Job - A job which does not require mounting a tape.

(5) Job Classification - Jobs are classified in three groups - debug (DEBUG), production (PRODb), and research computing (RECOM).

DEBUG indicates that a job is in the check-out phase.

PRODb refers to jobs which are in operation and require no programming changes.

RECOM covers those jobs in which the difficulties are not inherent in the actual program but rather in the computing techniques which are employed. This classification should be used when the program is past the debug stage and yet is not producing satisfactory results.

5.6.0 Nov/65 **Binary Deck Maintenance**

5.6.1 Nov/65 **Introduction**

To avoid burdening the user with the many details of handling his own binary decks, the policy of Ames has been to have these maintained at the central computer facility and requested by the user on his EOF card. The rules which governed this method of operation were:

a. A recompilation of a deck having the identical CAB number would replace any binary deck filed under that number.

b. Periodically, CAB operators with the approval of the users would purge the files, placing infrequently used programs in a different area.

c. The user could request several decks to be combined, assigned a different number and hence create a production deck.

Under DCS operation, the greatest deterrent to turnaround time has been the handling of binary decks. With the advent of the remote stations some provision had to be made to make general purpose subroutines available to the station as well as to keep their turnaround time to a minimum.

As a result of these problems, a method has been developed to place binary decks on the disk at the time of compilation and to update them according to the previous procedures. Techniques are provided for purging the disk of infrequently used decks, adding decks to the disk, and, most important, making available to the user a list of the decks which are on the disk. The details of how binary decks will be added to a job and how these other operations take place are described in the following sections.

5.6.2 Nov/65 **Use of Decks on Disk**

In order to bring the binary decks from the disk into the job the user supplies the necessary $DECK control cards with his job. For the description and placement of the $DECK control card see:

1) Section 6.2.1 for FORTRAN IV jobs.

2) Section 7.2.1 for FORTRAN II jobs.

3) Section 6.6.3 for OVERLAY jobs

A binary deck resulting from a compilation or an assembly is available to the user for another job 10 minutes after he receives the output from the first job.

5.6.3 Nov/65 **Binary Decks at Compile or Assembly Time**

Binary decks produced by a compile or an assembly are placed on the disk. A recompilation or a reassembly of a deck having the identical CAB number automatically replaces an existing deck on disk bearing that number.

5.6.4 Nov/65 **Purging Binary Decks From Disk**

Binary decks which belong to the FORTRAN IV or FORTRAN II Card Subroutine Library will always remain on disk. All other decks which have not been used for one month will be purged from the disk but the card copies remain on file. Decks will be purged weekly at which time a listing of binary decks is produced for distribution.

5.6.5 Nov/65 **Adding Binary Decks to Disk**

As mentioned earlier, any binary deck resulting from an assembly or a compile is automatically placed on disk. In addition a programmer may wish to reactivate a job whose decks have been purged from disk. To do so he furnishes the DCS operator with a list of the necessary decks. The DCS operator will place the decks on disk and will notify the programmer when it is done so that the job may be processed.

5.6.6 Nov/65 **List of Binary Decks on Disk**

Except for the binary decks which are on the FORTRAN IV subroutine library tape or on the FORTRAN II subroutine library tape, the binary decks used by programmers are on the 1301 disk. These decks include:

1) Those a user produces by a FORTRAN IV compile, a FORTRAN II compile, a COBOL compile, a MAP assembly or a FAP assembly

2) Those belonging to the FORTRAN IV Card Subroutine Library or the FORTRAN II Card Subroutine Library.

A listing is posted weekly giving information pertaining to all binary decks on disk. This listing contains:

1) A sorted list of the binary decks available on the disk.

2) The date when each binary deck was placed on the disk.

3) The date when each binary deck was last used (loaded) from the disk.

4) A list of all decks which have been purged because of non-use.

Any deck listed as available remains available at least until a new listing is distributed. A copy of this listing will be sent to each remote station and one will also be posted in room 129 at the Data Reduction Building.

5.6.7 Nov/65 **Production Decks and Disk**

Having binary decks on disk eliminates the need of maintaining Production decks and therefore no new Production decks will be placed on disk. The user can easily maintain a set of $DECK Cards as a Production deck.

5.6.7 Nov/65 (continued)

Any existing Production deck requested by a programmer will be placed on disk. The operators will place the deck with a P as the last letter in the CAB number so that F02490 would then be F0249P. Any production deck that needs to be modified will be destroyed and the programmer will then ask for individual binary decks.

5.6.8 Nov/65 Card Copies of Binary Decks

Card copies of binary decks are filed 24 hours after they are produced and are then available for the programmer's use. The card copies are necessary only for:

(1) Job transmittals.

(2) Reactivating jobs whose binary decks have been purged from disk.

5.7.0 Nov/65 **Operating System Restrictions**

5.7.1 Nov/65 **Use of Sense Switches on the 7094**

The six sense switches on the 7094 are not available for the programmer's use. They are reserved for the operating system.

5.7.2 Nov/65 **Use of PAUSE, HLT, HPR on the 7094**

It is not possible for the programmer to stop the 7094 in order to give instructions to the DCS operator, and so the FORTRAN statement PAUSE, and the MAP instructions HLT, HPR should not be used.

5.7.3 Nov/65 **Mounting of Tapes**

A maximum of 6 tapes can be mounted for a particular job.

5.7.4 Nov/65 **FORTRAN II Chain Jobs**

Chain tapes must be used on tape units B2, or, preferably, B3.

6.0.0 Jul/65 IBJOB MONITOR SYSTEM

6.1.0 Jul/65 Function

IBJOB is the control monitor for the FORTRAN IV compiler, the COBOL compiler (a business oriented language), MAP (the Macro Assembly Program), and the DEBUGGING PACKAGE. This monitor also handles loading of the binary decks (IBLDR) and searches the library for necessary subroutines (IBLIB). Control cards specify to the monitor the action to be performed.

6.2.0 Jul/65 FORTRAN IV

6.2.1 Nov/65 Job-Makeup

A FORTRAN IV job which uses the DEBUGGING PACKAGE for either post-mortem dumping or dynamic dumping has the following composition:

(a) PLOT card (solid blue)

(b) EOF card (striped color/category)

(c) $JOB card (solid orange)

(d) $SETUP cards (solid orange), if tapes are to be mounted or printed

(e) $IBJOB card (solid orange)

(f) $IBDBL card (solid red)

(g) *DEBUG card (solid red)

(h) *ETC card (solid red)

(i) DEBUGGING REQUESTS (solid red)

Items (g), (h), and (i) may be repeated as often as necessary.

(j) *DEND card (solid red)

(k) $IBFTC card (solid orange), if FORTRAN IV compile

(l) FORTRAN IV source deck (grey stripe)

(m) END card (brown stripe)

Items (k), (l), and (m) may be repeated as often as necessary.

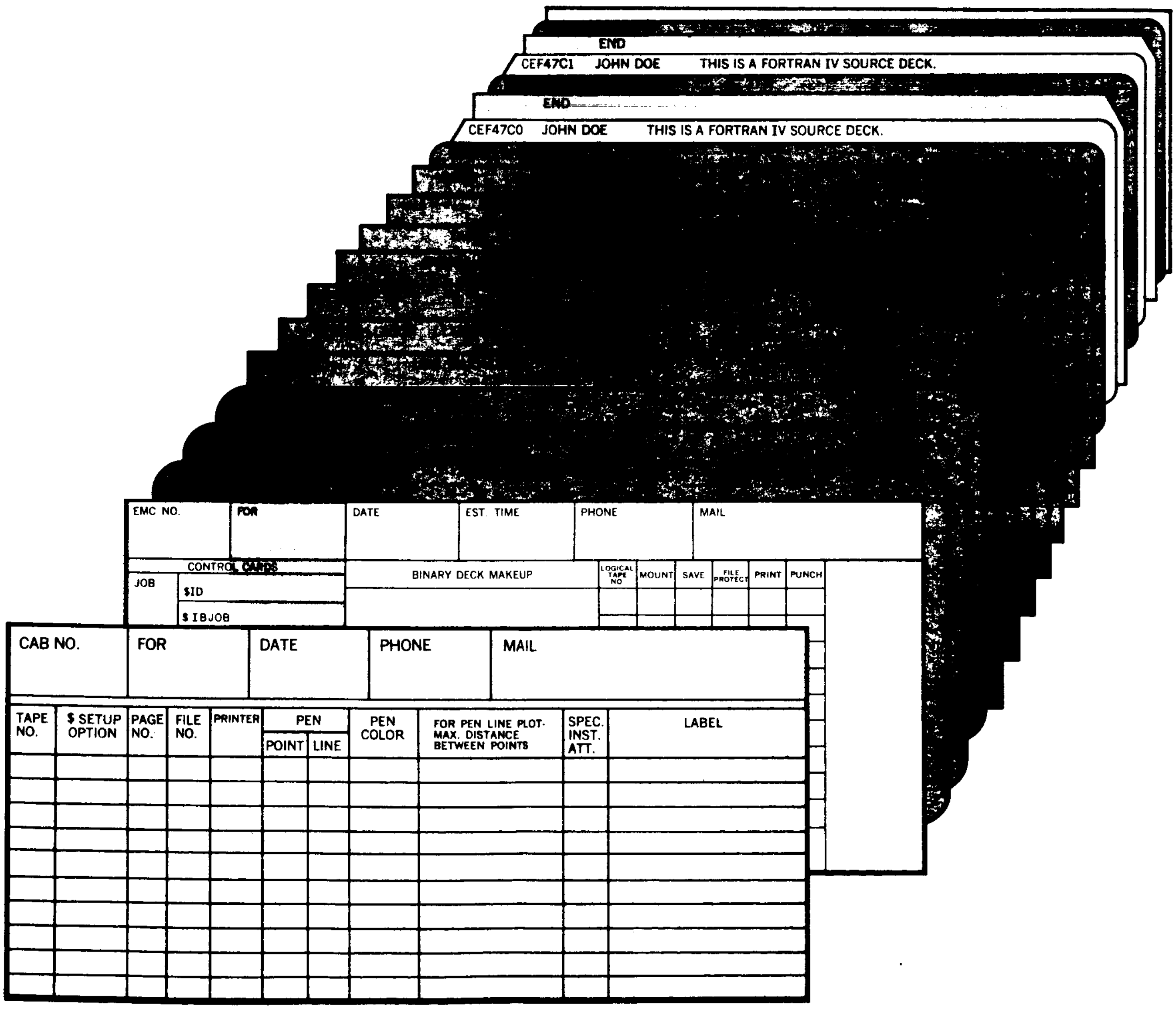
→ (n) $DECK cards (solid orange)

(o) $DATA card (solid orange)

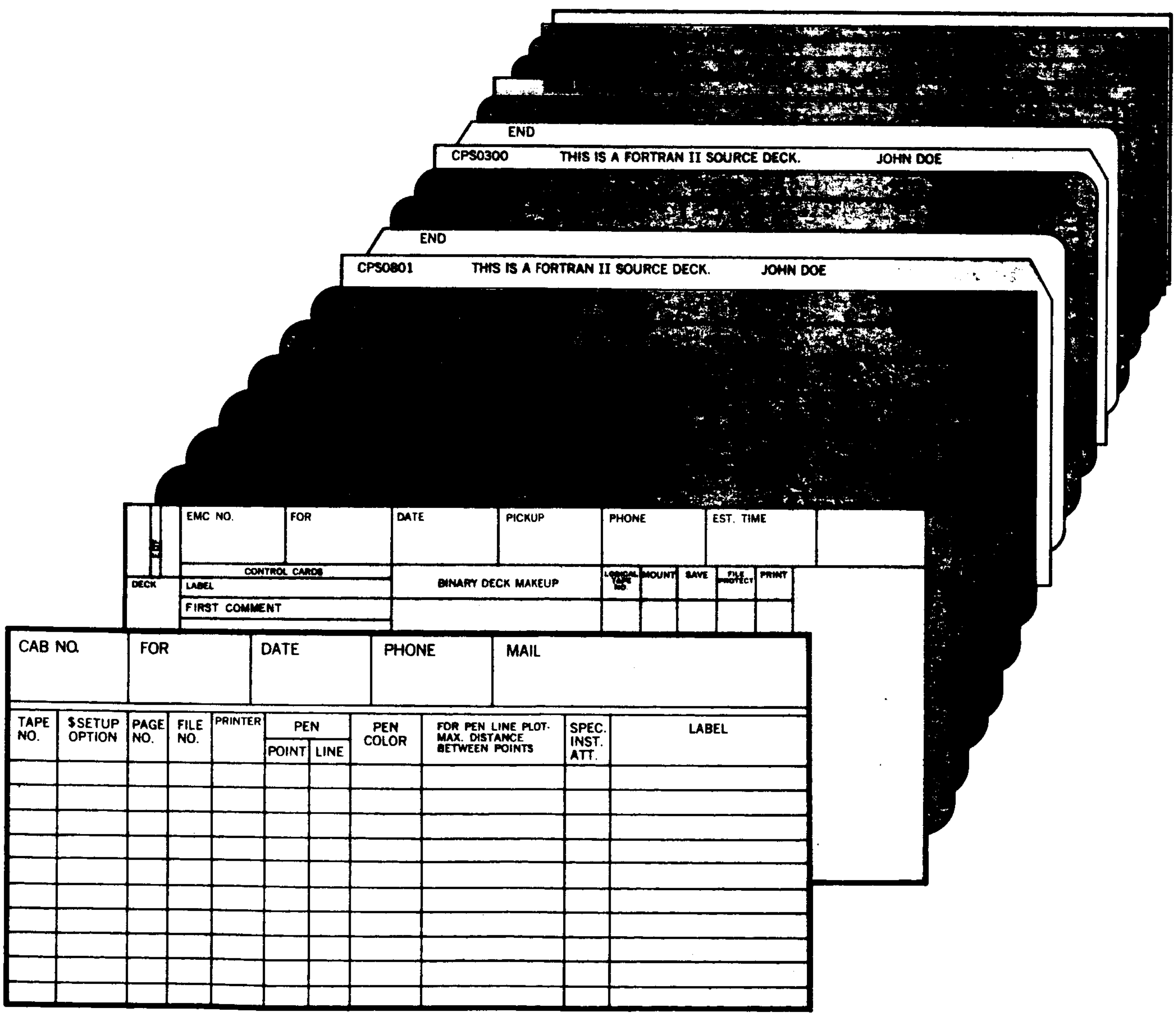
(p) INPUT DATA cards, if required (any color except solid orange; FORTRAN cards should not be used)

6.2.2 **Sample Job Deck**
Jul/65

Sample Fortran IV Job



Sample Fortran II Job



6.2.3 Jul/65 <u>Control Cards</u>

6.2.3.1 Jul/65 <u>Plot Card</u> (solid blue)

This card is used first by the DCS operator and later by the EAI Dataplotter operator. It is the first card of every job which writes a plot tape. The following is an example showing its usage:

[redacted] NO. CAB	FOR	DATE	[redacted]	PHONE	MAIL
EF18YO	John Doe	6/30/65		2895	

TAPE NO.	$SETUP OPTION	PAGE NO.	FILE NO.	PRINTER	PEN POINT	PEN LINE	PEN COLOR	FOR PEN LINE PLOT-MAX. DISTANCE BETWEEN POINTS	SPEC. INST. ATT.	LABEL
7	Plot 1	1	1	✓		✓	Red	2.0		IDVSVI Study, Subjs. 1-12, 16,
			2			✓	Red	0.25		Table 1
			3			✓	Green	0.25		
		2	4	✓		✓	Red	2.0		IDVSVI Study, Subjs. 1-12, 16,
			5		✓		Red			Table 2
9	Plot 2	3	6	✓		✓	Red	2.0		IDVS2 Study, Subjs. 1-6,
			7			✓	Red	0.25		Tables 1 & 2
			8			✓	Green	0.25		
			9		✓		Red			

For a detailed explanation of this card see the section on PLOTTER USAGE (8.5.0).

6.2.3.2 Nov/65 <u>EOF Card</u> (striped color/category)

This control card provides information for the operator. It follows the blue Plot card and is prepunched with a 7 and 8 in column one. The color of the vertical stripe on the card identifies the category of the job in the following way:

No vertical stripe (solid white) - Standard Monitor DEBUG or RECOM Job

Yellow vertical stripe - Standard Monitor PRODb Job

Brown vertical stripe - Special Monitor DEBUG or RECOM Job

Green vertical stripe - Special Monitor PRODb Job

Red vertical stripe - Priority Job (available from S. M. Crandall at CAB)

A new card must be provided with each job and it will not be returned. It should contain:

1) CAB number for the job. (This must be identical to the CAB number on the $JOB card).

2) Name of the programmer.

3) Date of submission of the job.

4) Estimated execution time.

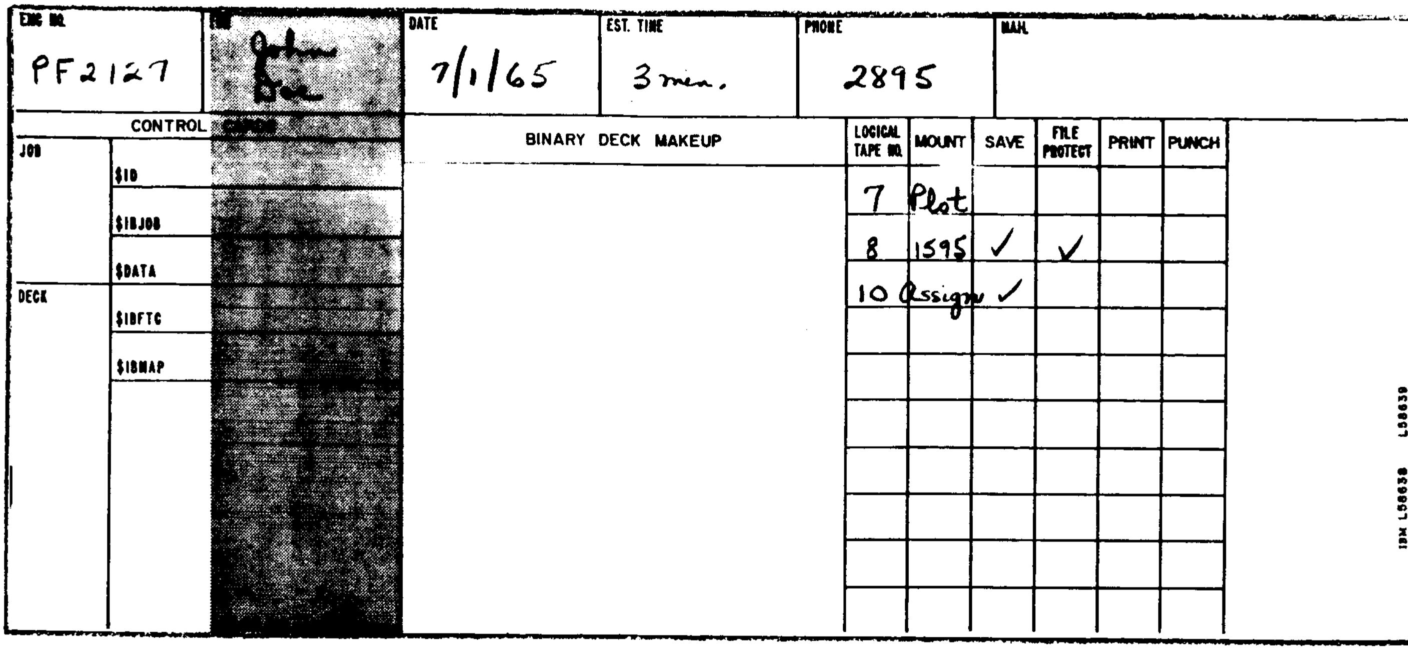
5) Number at which the user can be contacted.

6) Mail stop to which the user wishes the data returned.

→ 7) Additional tape assignments other than 5 for input, 6 for output, scratch tapes, and print tapes. (See Magnetic Tape Usage, section 8.4.0)

In addition, this card supplies the user with a checklist for deck makeup. The blank column at the right of the card may be used for remarks

6.2.3.2 (continued)
Nov/65



EMP NO.	NAME	DATE	EST. TIME	PHONE	MAN.
PF2127	John Doe	7/1/65	3 min.	2895	

CONTROL CARDS		BINARY DECK MAKEUP	LOGICAL TAPE NO.	MOUNT	SAVE	FILE PROTECT	PRINT	PUNCH
JOB	$ID		7	Plot				
	$IBJOB		8	1595	✓	✓		
	$DATA		10	Assign	✓			
DECK	$IBFTC							
	$IBMAP							

6.2.3.3 <u>$JOB Card</u> (solid orange)
Jul/65

This control card is used by the IBSYS monitor and follows the EOF card. If for any reason it is not present, the job will be deleted. It contains information necessary for job control and accounting and has the following format:

Column	Contents
1-4	$JOB
8-12	Job Classification (DEBUG, RECOM, PRODb)
13-24	Name of Research Scientist
25-30	CAB Number for the Job
31-35	Job Order Number
36	Blank
37-42	Estimated Lines of Output, right-justified
43-48	Estimated Execution Time in minutes and hundredths of a minute. The decimal point must be in column 46.
49-76	Available for the User
77-80	Phone Number of User

6.2.3.3 (continued)
Jul/65

Notes on Above

1) Job Classification - All words are left justified, i.e., DEBUG, RECOM, PRODb.

2) Name - This may go anywhere in the field but to aid accounting procedures, each user is urged to adopt a single identifying name for his work. For example, if a person uses both J. A. Doe and John Doe, the accounting records will assume there are two separate individuals.

3) CAB Number - This is a six-character alpha-numeric method of coding jobs for identification purposes. As assigned by CAB, it is composed of two letters which identify the branch, two numbers and two zeros, i.e., TA2400. To identify decks within the job the two zeros may be replaced by any number or letter of the programmer's choice, i.e., TA2497, TA24AZ. When a job is being processed by DCMUP, the DCS operators can identify it only by its CAB number. If a rerun is necessary, the DCS operators must be able to correlate console messages and EOF cards with $JOB cards. It is therefore necessary that all jobs which conceivably could be in the input or output queue at the same time have unique CAB numbers on the $JOB card (i.e., TA2400, TA2410, TA2420). The CAB number written on the EOF card must agree with the number on the $JOB card.

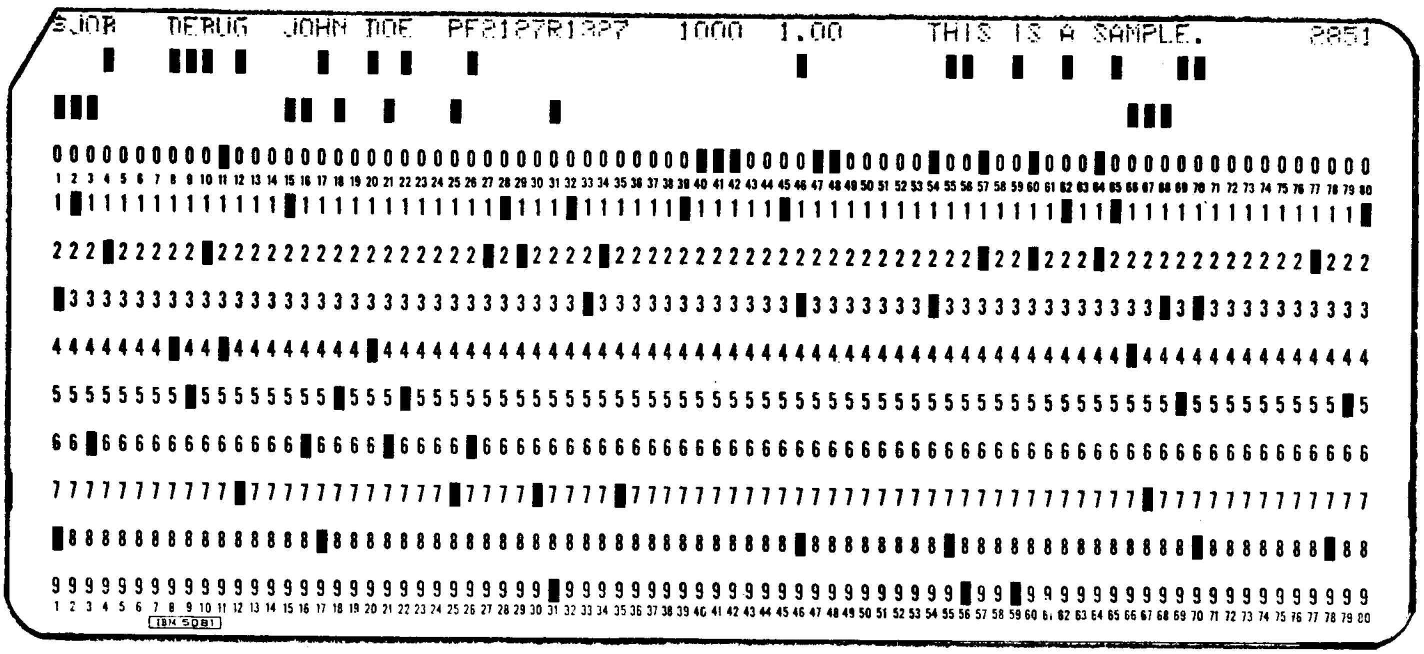
4) Job Order Number - This number must be punched in the following form:

   ANNNN, where A is an alphabetic field and N is a numeric field. The numeric field must be right-justified. (Any column may be left blank as long as the above format is maintained.) If any job order number does not fit this format, please contact 7094 Operations for an adjustment.

5) Estimated Lines of Output - This is an estimate of the number of lines of execution output a given job will produce. This number will be entered into a checking routine on the 7040, and when it is exceeded, execution of the job will be terminated. One should also keep in mind that position in the input queue is determined by the number of lines of output requested. If columns 37-42 are left blank, then the system will determine the maximum allowable lines, which currently is 5000.

6) Estimated Execution Time - This is the time the job is expected to execute on the 7094. This is only object execution time, not compilation time. Here again, if the estimate is exceeded, execution will be terminated. If no execution time is specified, the system will determine the maximum allowable time, which currently is 5 minutes for DEBUG and RECOM jobs and 10 minutes for PRODb jobs.

6.2.3.3 (continued)
Jul/65



6.2.3.4 <u>$SETUP Card</u> (solid orange)
Jul/65

This control card is used by the 7040 (DCMUP) and follows the $JOB card. A $SETUP card is required for all special tapes a program uses. Input tape 5, output tape 6, and scratch tapes do not require setup cards. All tapes to be processed by DCS must be in the standard DCS record format. This format consists of 460-word physical records. The reason for this format is that a disk track contains 460 words of storage, and DCMUP reads and/or writes one track of information at a time. To provide I/O optimization DCMUP has set up a number of 460 word buffers which must be used for tape files as well as disk files. It is therefore necessary that all tapes used as input be blocked to 460 word records. All output tapes are blocked to 460 word records and may be deblocked if necessary.

The primary functions of the $SETUP card are:

1) To inform the 7040 (DCMUP) that the specified reel of tape is to be processed by the 7094 program so that messages to the operator can be constructed and the tape reel mounted in preparation for execution.

2) To allow the programmer to specify desired preprocessing and post-processing utilities, such as blocking input tapes, deblocking output tapes, and printing disk file information. The $SETUP card has the following format:

Column	Contents
1-6	$SETUP
8-9	UNIT
16-72	VARIABLE FIELD Option 1, Option 2, File Count b Comment

Notes on Above

1) UNIT - This is the logical tape number used by the program and may be 7, 9, or 11 for BCD tapes and 8, 10, or 12 for binary tapes.

2) Option 1 and Option 2 have the following functions:

   (a) Specify the type of input/output device (tape, disk) that will be assigned to functions normally performed by 'UNIT'.

   (b) Provide the system with blocking and deblocking requirements of the job.

   (c) Provide information to the system to be used in messages to the operator concerning tape handling.

   The following may be used as options:

Option 1	Option 2
1. A tape number	1. A tape number
2. ASSIGN	2. ASSIGN
3. DISK	3. PLOT
	4. DISK
	5. PRINT
	6. NORING

1. A tape number refers to a permanent tape which has a previously assigned number.

2. ASSIGN is used when a new tape is created for which a permanent tape number is to be assigned. For identification purposes, if a job is creating more than one new permanent tape, ASSIGN may be replaced by ASIGN1, ASIGN2, etc.

3. PLOT is used for plot tapes. PLOT1, PLOT2, etc. may replace PLOT when a job has more than one plot tape.

4. PRINT is used when the programmer wishes to print output which he has written on a tape other than tape 6.

5. NORING is used when a permanent input tape is file protected to prevent writing on the tape.

Option 1 specifies the first input/output device for the data the program is processing. Option 2 names the second device when such is necessary.

3) File Count. The number of files to be blocked or deblocked must be specified. It is safe however to overspecify the file count for a tape, which is to be deblocked and which the program has just written, since the 7040 will deblock either the count specified or the number of files just written, whichever is smaller. If File Count is omitted, one file will be assumed.

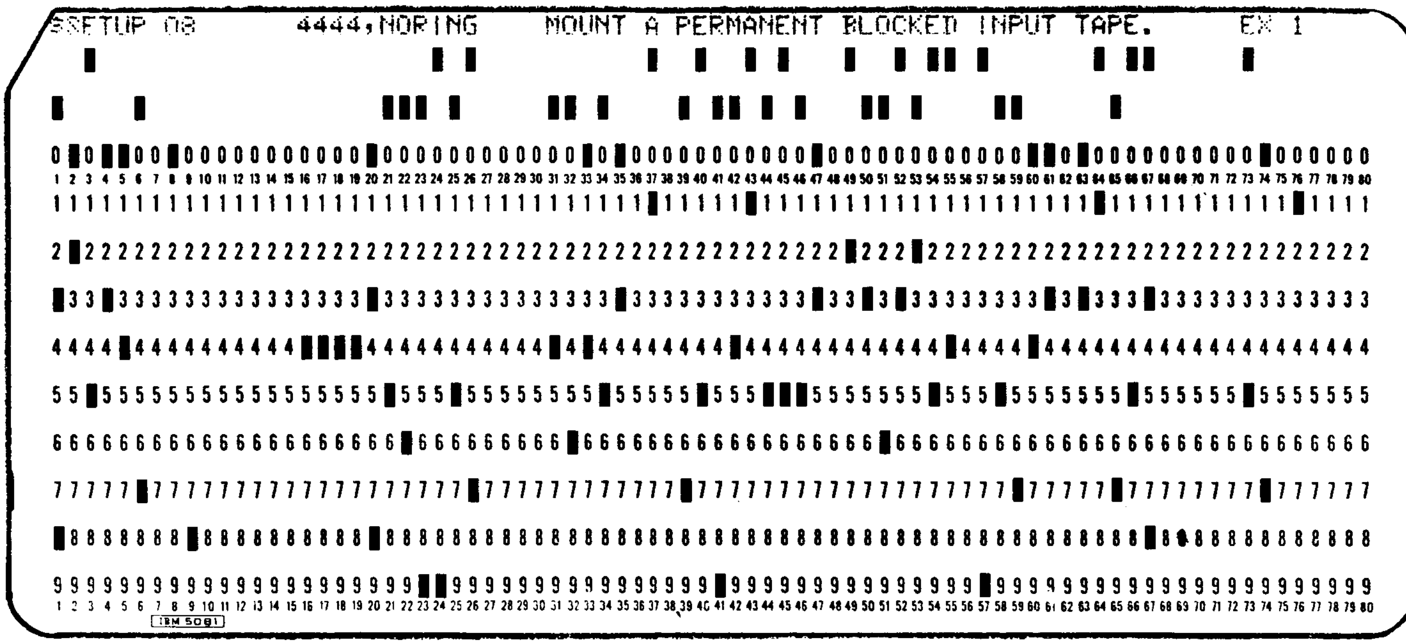
4) When a blank is encountered in the VARIABLE FIELD, the remainder of the card is treated as a comment.

The options and file count are used as follows: (Also see examples which follow.)

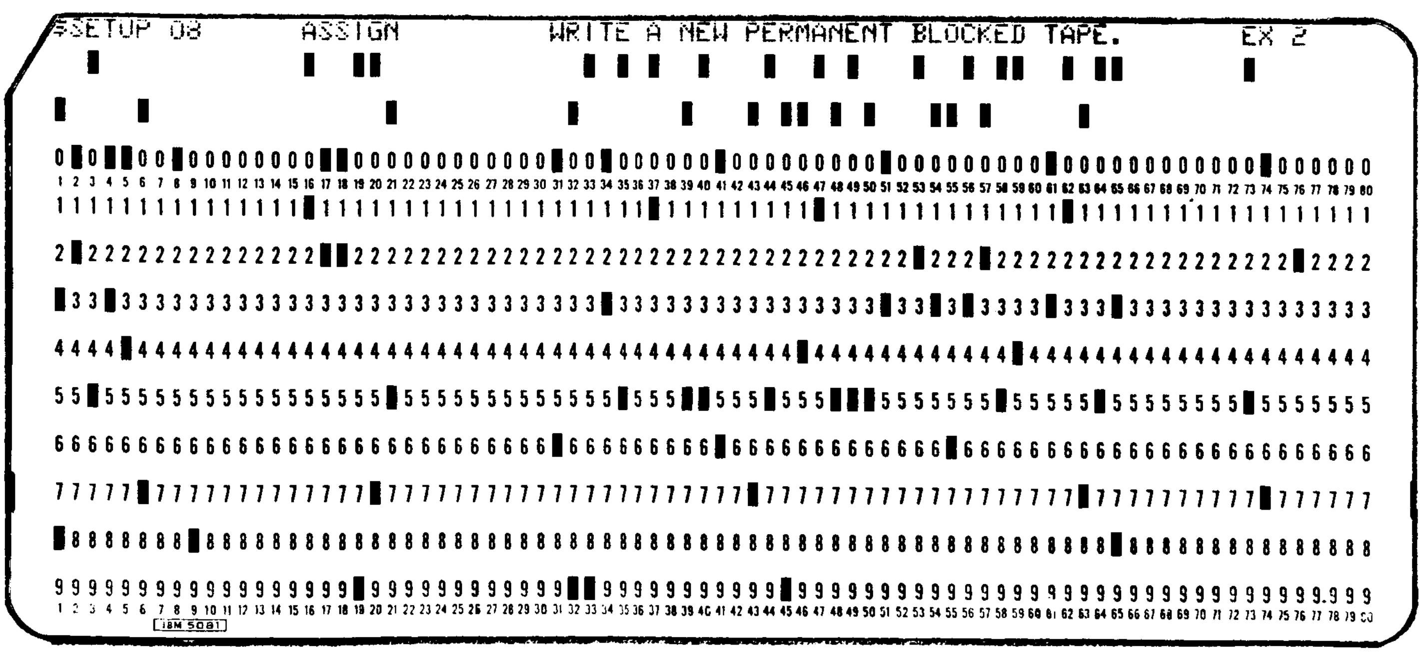
Option 1	Option 2	File Count	Function
1. a tape number	NORING	blank	Mount a permanent blocked tape which is to be used as input only and is file protected.
2. ASSIGN	blank	blank	Write a new permanent blocked tape.
3. a tape number	blank	blank	Write on a permanent blocked tape.
4. DISK	PRINT	specify	Print output written on a tape other than tape 6.
5. DISK	PLOT	specify	Write a tape to be plotted.
6. DISK	ASSIGN	specify	1) Write a tape to punch cards off-line. 2) Write a tape for transmittal. 3) Write a permanent unblocked tape.
7.a tape number	ASSIGN	specify	Make a permanent blocked copy of an unblocked tape.
8. a tape number	DISK	specify	Information is to be used from an unblocked tape, but no blocked copy of the tape is desired.

6.2.3.4 (continued)
Jul/65

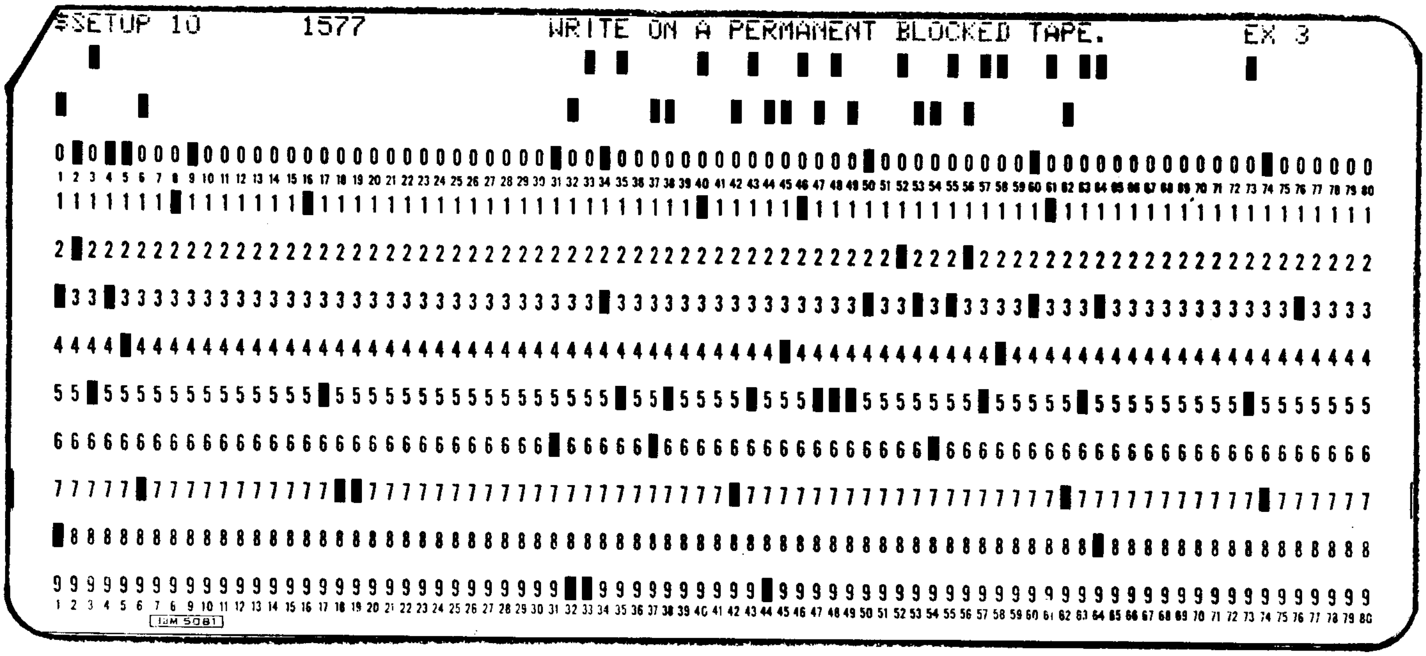
Examples:



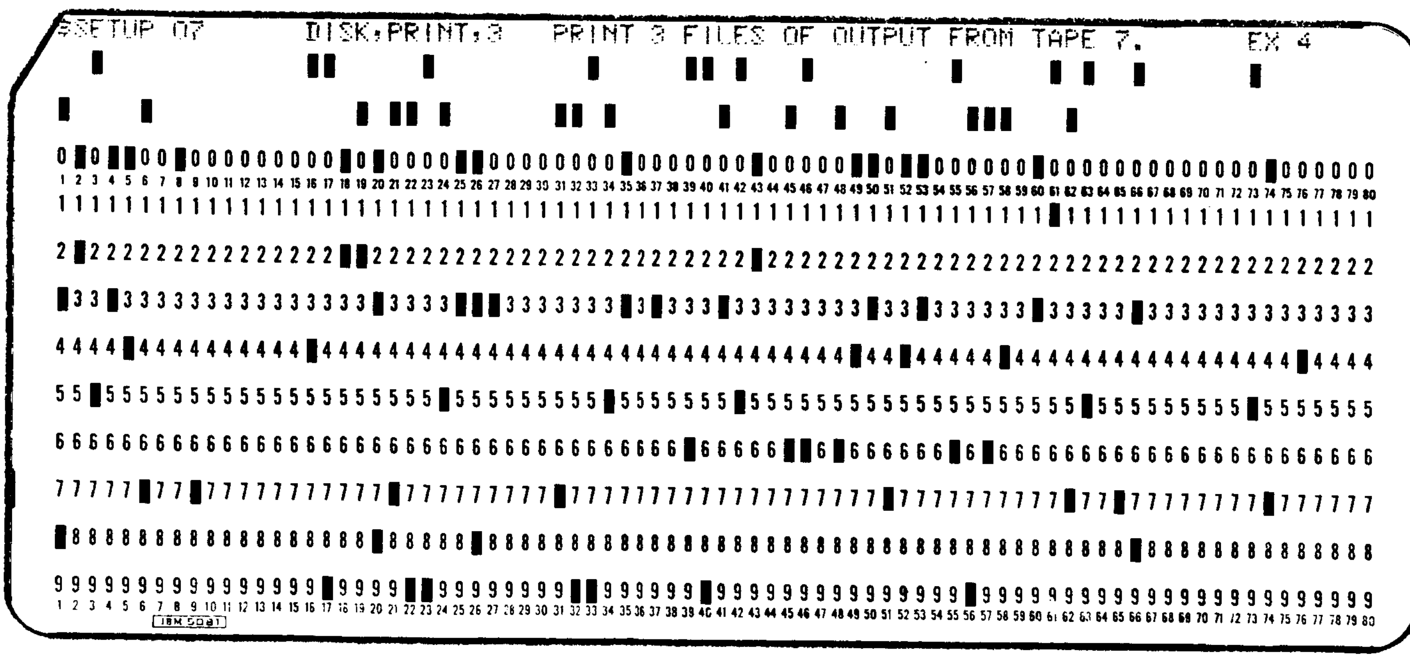
Explanation: The programmer's permanent tape 4444 is mounted as tape 08 and is to be used as an input tape. NORING indicates it cannot be written on.



Explanation: The programmer wants to create a new blocked tape which is to be assigned a permanent tape number.

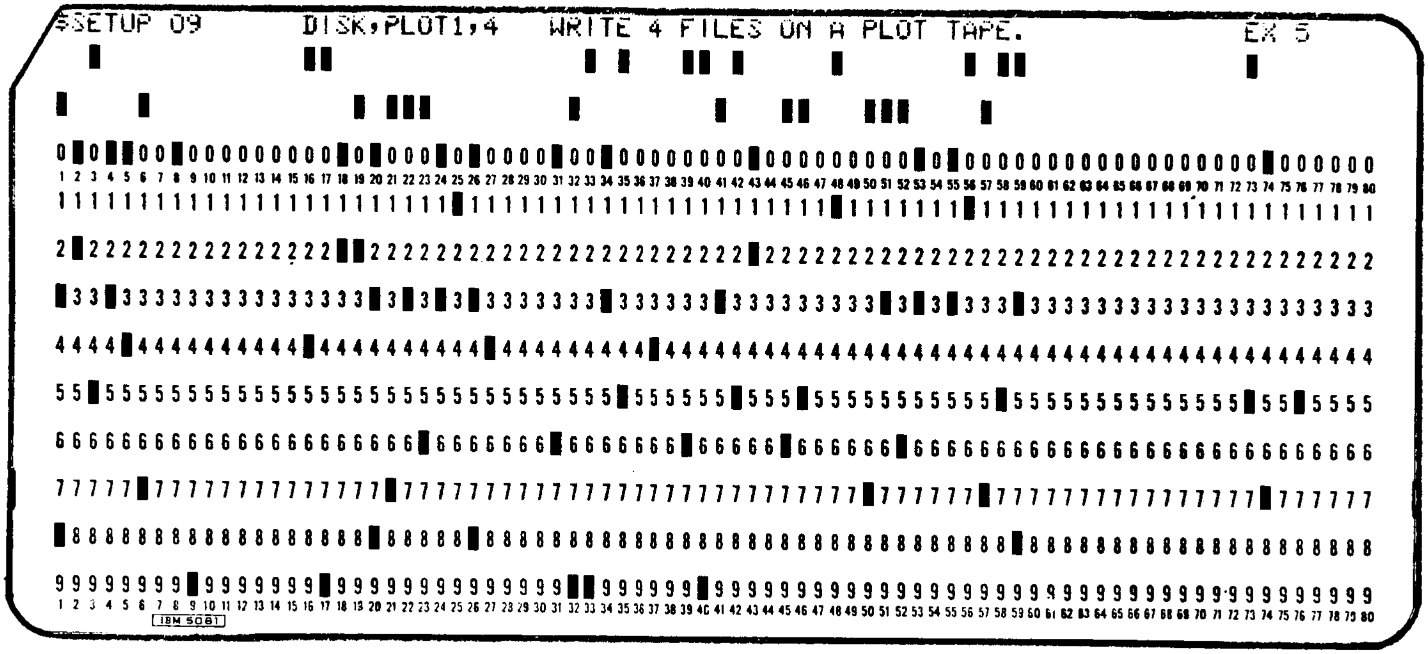


Explanation: The programmer wishes to write on his permanent tape 1577.

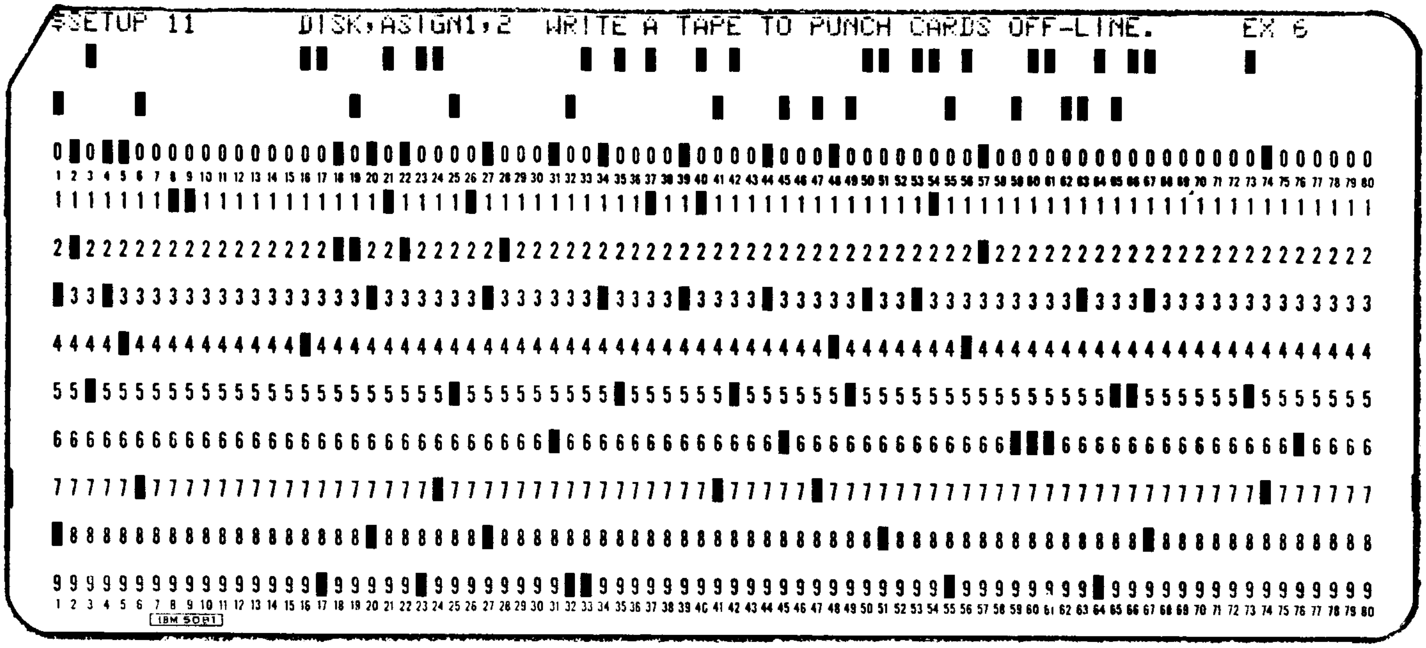


Explanation: Three files of output, intended for tape 7, are written on the disk during the processing phase and printed on the 1403 during the postprocessing phase.

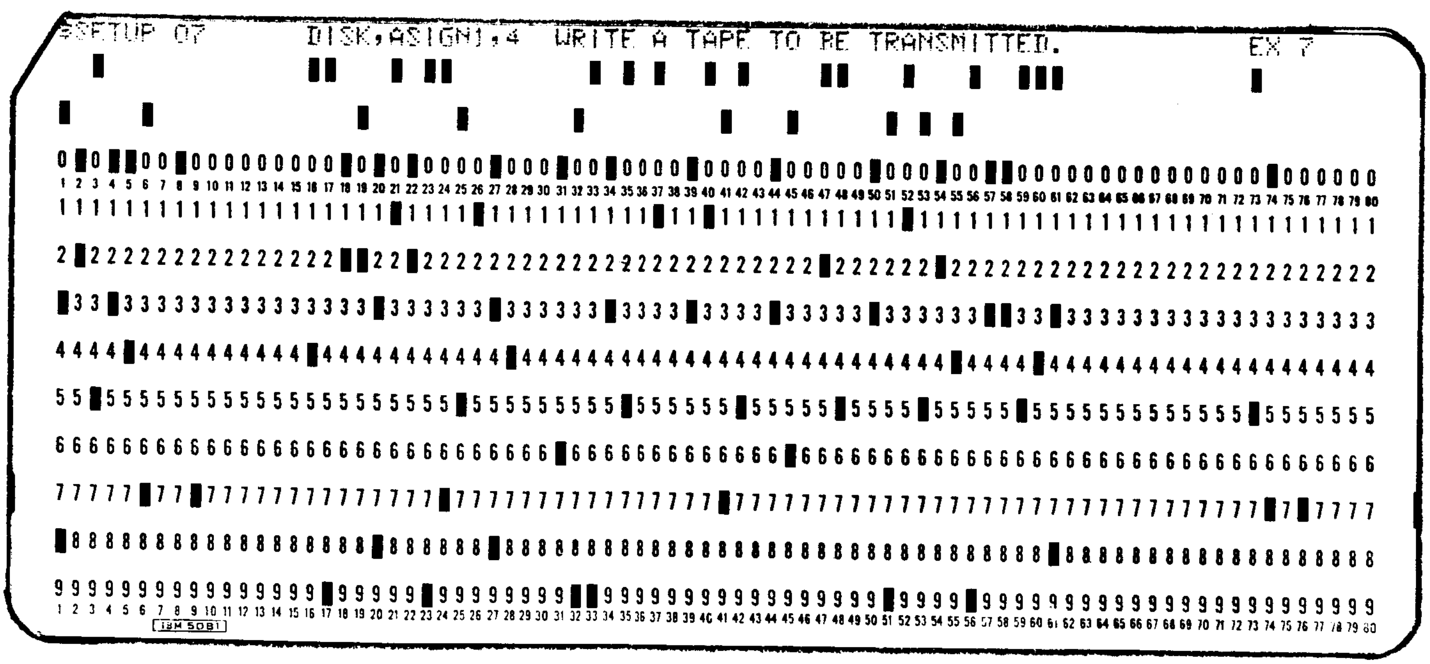
6.2.3.4 (continued)
Jul/65



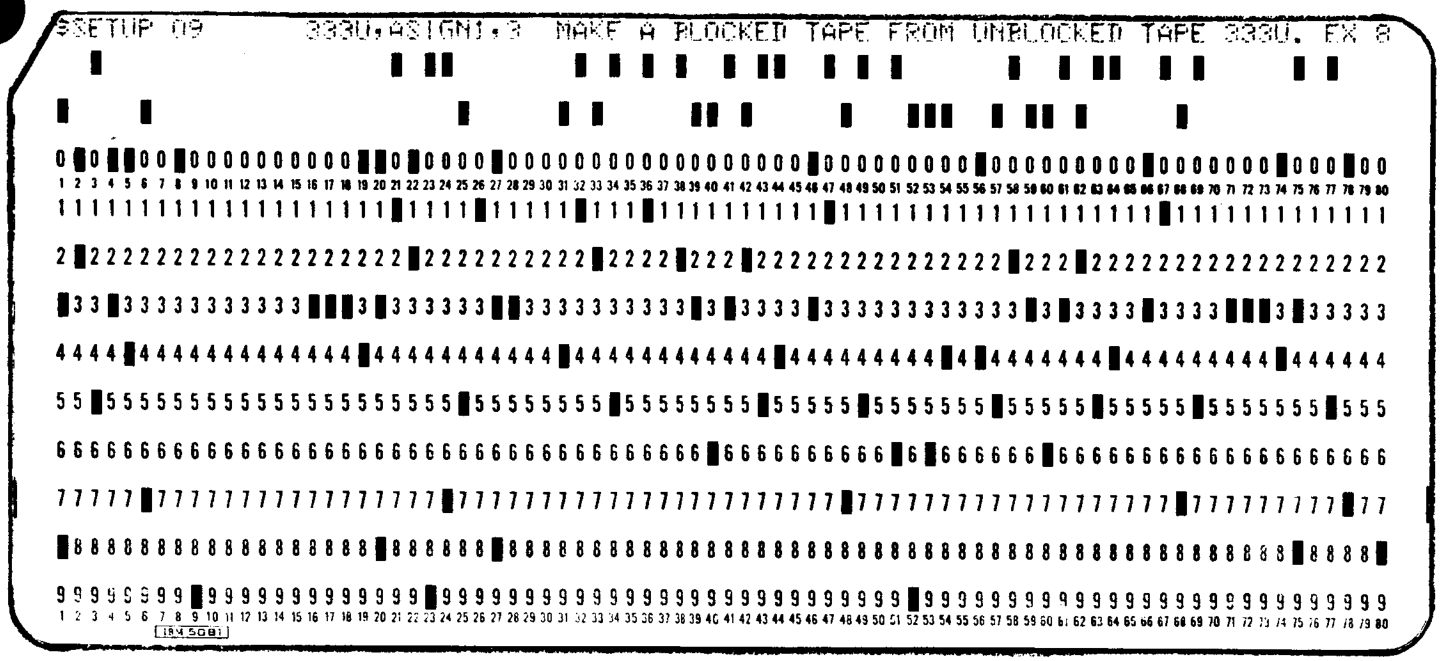
Explanation: Plotting data intended for tape 9 are written on the disk during the processing phase. During the postprocessing phase, 4 files will be deblocked and written on a plot tape. (Note: It is also necessary that a blue plot card, correctly filled out, precede the EOF card.)



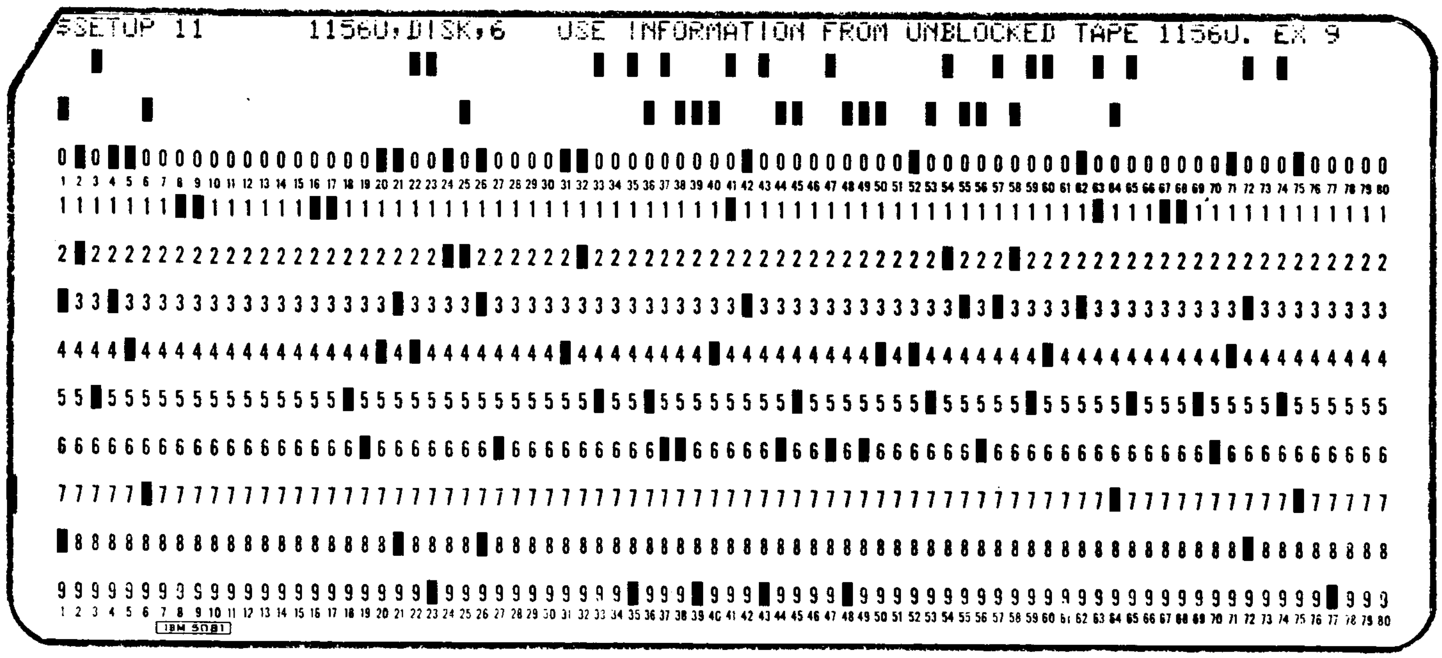
Explanation: Punch data intended for tape 11 are written on the disk during the processing phase. During the post-processing phase, 2 files will be deblocked and written on a tape which will be assigned a number. The tape is then processed on a 1401 to produce the actual cards. (Note: It is necessary for the programmer to specify on the EOF card (1) the number of files to be punched, (2) the approximate number of cards, and (3) the card color.)



Explanation: Data intended for tape 7 are written on the disk during the processing phase. During the postprocessing phase, 4 files will be deblocked onto a tape which will be assigned a number.



Explanation: During the preprocessing phase unblocked tape 333U is blocked to the standard DCS record format and written on a permanent tape, referred to as tape 09 in the user's program. This tape will be assigned a number.



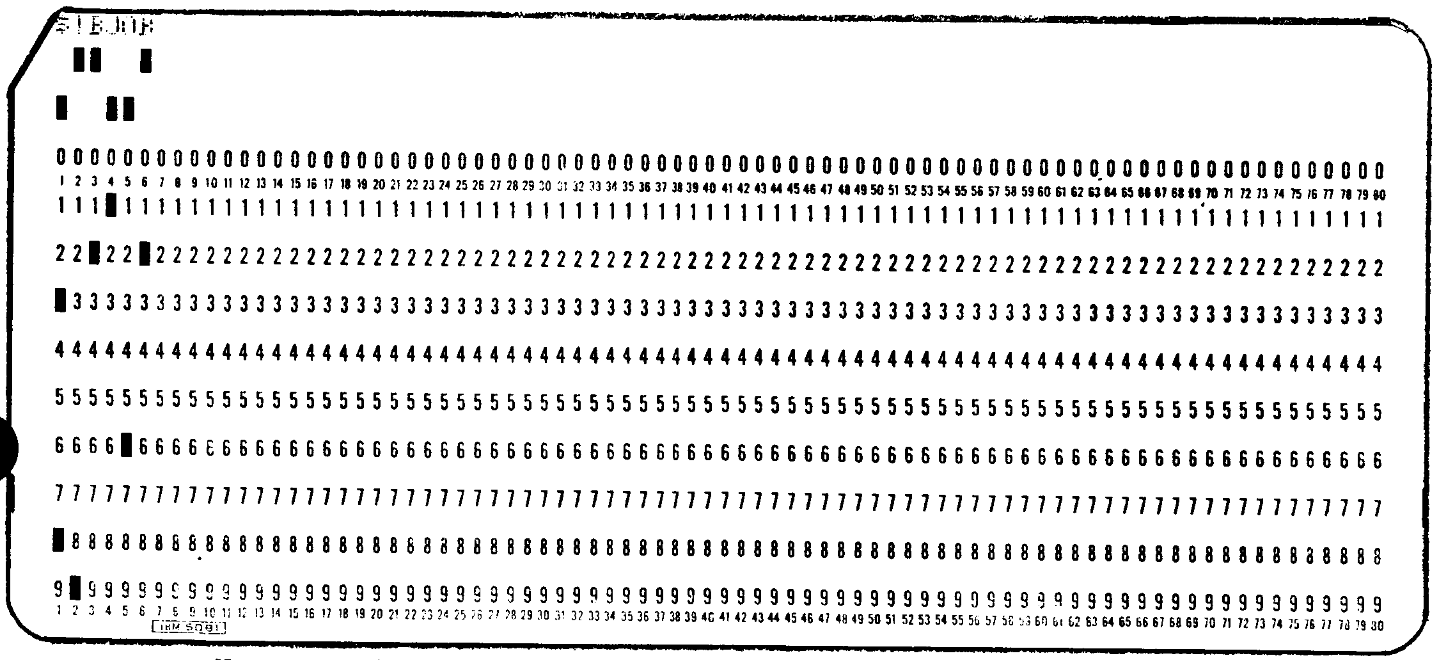
Explanation: During the preprocessing phase 6 files of data from the unblocked tape, 1156U, are blocked to the standard DCS record format and written onto disk. The data are referred to as tape 11 by the program. A blocked tape is not produced.

6.2.3.5 Nov/65

$IBJOB Card (solid orange)

It must be recalled that IBSYS itself selects monitors which then control specific jobs. It is therefore necessary to tell the IBSYS monitor the type of job it is to execute. In the case of FORTRAN IV, the controlling monitor is IBJOB, and the required control card format is:

Column	Contents
1-6	$IBJOB
16	Options



Notes on Above

This control card follows the $SETUP card(s). Since this card specifies the options which are executed as this job is processed, some choices are available for the programmer. These are outlined below:

1) If no options are specified on the $IBJOB card, then the job
   (a) executes, if possible,
   (b) provides a storage map at load time, and
   (c) contains some decks for compilation.

2) If the programmer does not wish this job to execute even if it does load, then his $IBJOB card has the following format:

Column	Contents
1-6	$IBJOB
16-19	NOGO

6.2.3.5 (continued)
Nov/65

3) If the programmer does not have any FORTRAN decks or Debug Request cards in his job, he can reduce the loading time by specifying a NOSOURCE option. In that case, the $IBJOB card has the following format:

Column	Contents
1-6	$IBJOB
16-23	NOSOURCE

Prepunched no option versions of this card are available in DCS Operations area.

6.2.3.6 $IBDBL Card (solid red)
Jul/65

This card is used by the IBJOB monitor to load that part of the DEBUGGING PACKAGE which allows load time debugging. It is placed immediately following the $IBJOB card. Details concerning usage of the DEBUGGING PACKAGE can be found in a later section of this manual. The format for the $IBDBL card is:

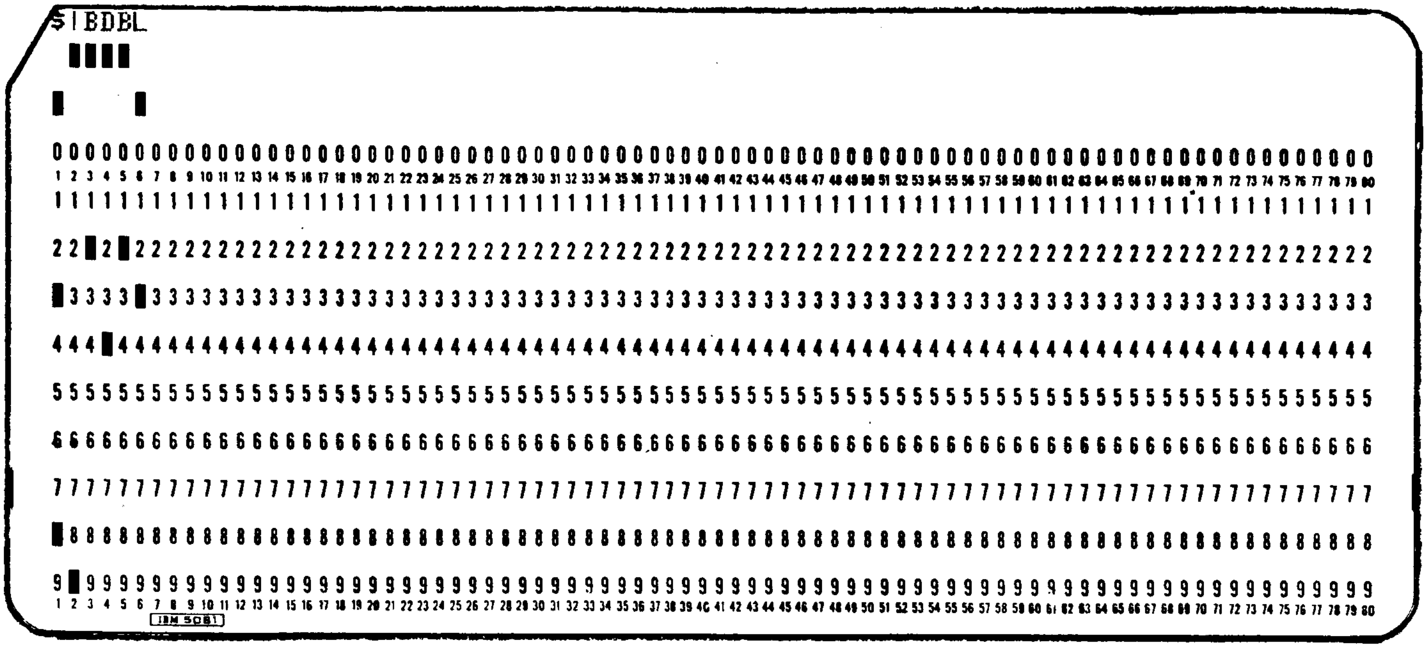
Column	Contents
1-6	$IBDBL
16-72	OPTIONS TRAP MAX = n_1, LINE MAX = n_2

where n_1 and n_2 are integers in the range 1 to 32767 with no leading 0. Either or both options in any order may be used to control the amount of debugging output, as follows:

TRAP MAX = n_1 Causes all debugging action to cease after n_1 requests have been executed. Every debug request encountered is counted irrespective of any resultant action.

LINE MAX = n_2 Causes dumping to cease after approximately n_2 lines of debugging output have been written.

Note: Blanks are allowed in this field since columns 16-72 are scanned in full for information.



Notes on Above

1) If no options are specified by the programmer, all debugging action ceases after 1000 lines of debugging output have been written.

2) If the programmer wishes a maximum of 2000 lines of debugging output, he may use the following format:

Column	Contents
1-6	$IBDBL
16-72	LINE MAX = 2000

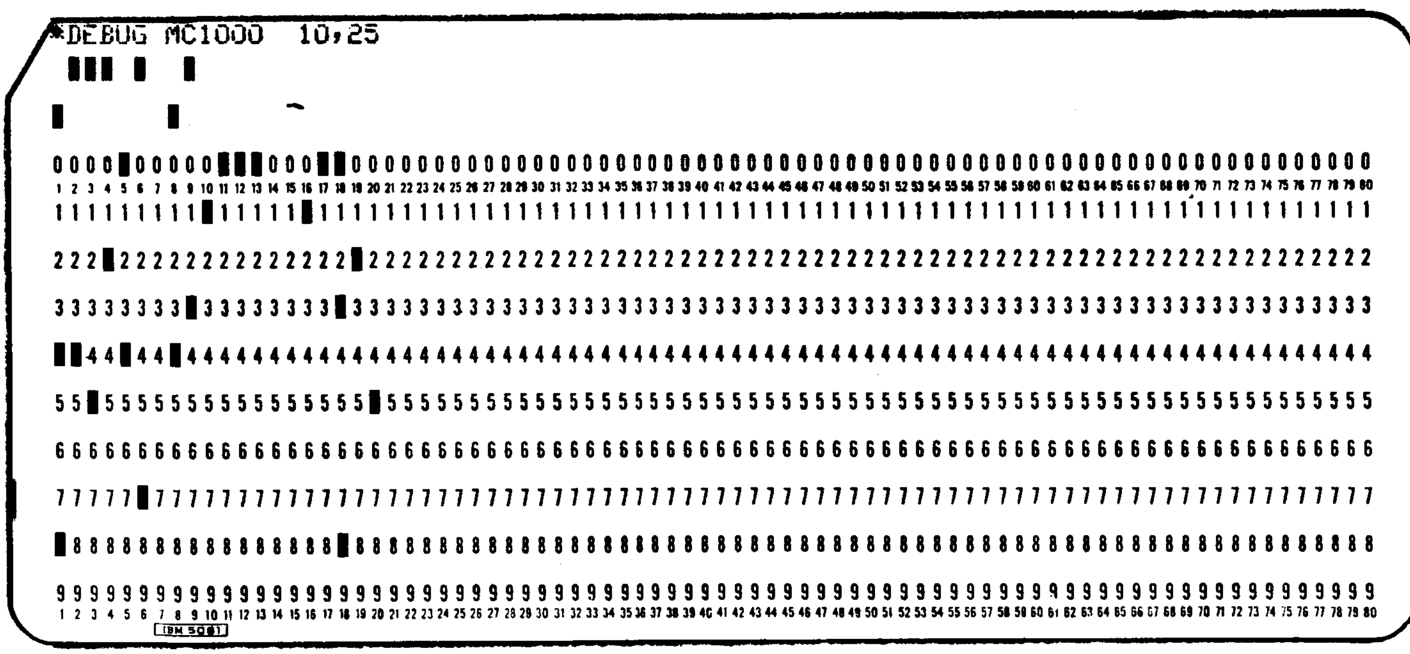
3) If the programmer wishes a maximum of 2000 lines of debugging output and in addition wishes debugging action to cease after 50 debugging requests have been executed, he may use the following format:

Column	Contents
1-6	$IBDBL
16-72	LINE MAX = 2000, TRAP MAX = 50

6.2.3.7 Jul/65 **<u>*DEBUG Card</u>** (solid red)

This control card is used by the DEBUGGING PACKAGE and specifies in which deck, as well as where in that deck, the debugging requests are to occur. The *DEBUG card has the following format:

Column	Content
1-6	*DEBUG
8-13	DECKNAME (CAB number)
16-72	VARIABLE FIELD loc1, loc2, loc3, ...



Notes on Above

DECKNAME - This is .LXCON for post-mortem dumps. For dynamic dumps it is the CAB number of the object deck (AB0100, AB0105, AB0109) to which this request applies. If this field is blank, the last DECKNAME specified on the preceding *DEBUG card is assumed; if a DECKNAME was not previously specified, the request is deleted.

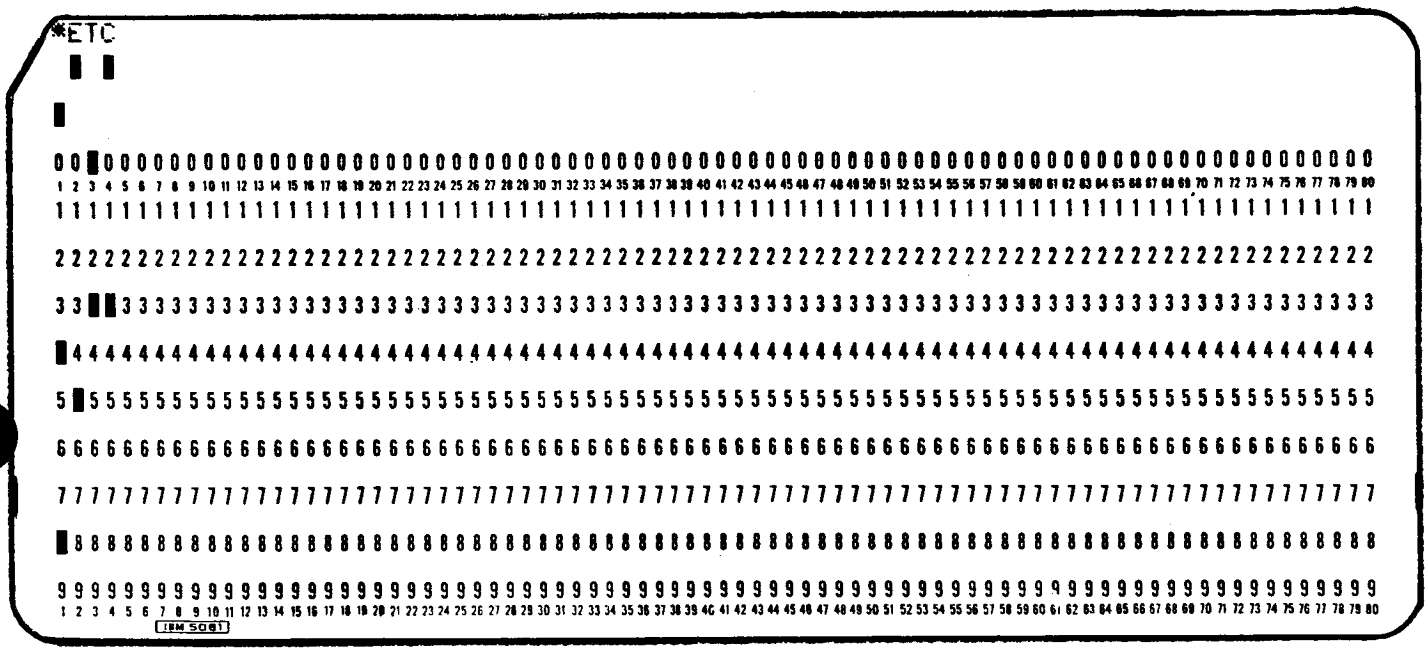
loc1, ... This field refers to the statement number(s) of executable statement(s) at which the debug request is to be inserted. It can also refer to subroutine entry points.

Note: Blanks may be included in the variable field for legibility, but they may not be imbedded, i.e., statement number 30 cannot be written as 3b0.

6.2.3.8 Jul/65 **<u>*ETC Card</u>** (solid red)

This control card is used to extend the variable field of the *DEBUG card, and as many as are necessary may be used. The *ETC card has the following format:

Column	Contents
1-4	*ETC
16-72	extension of the variable field of the *DEBUG card.

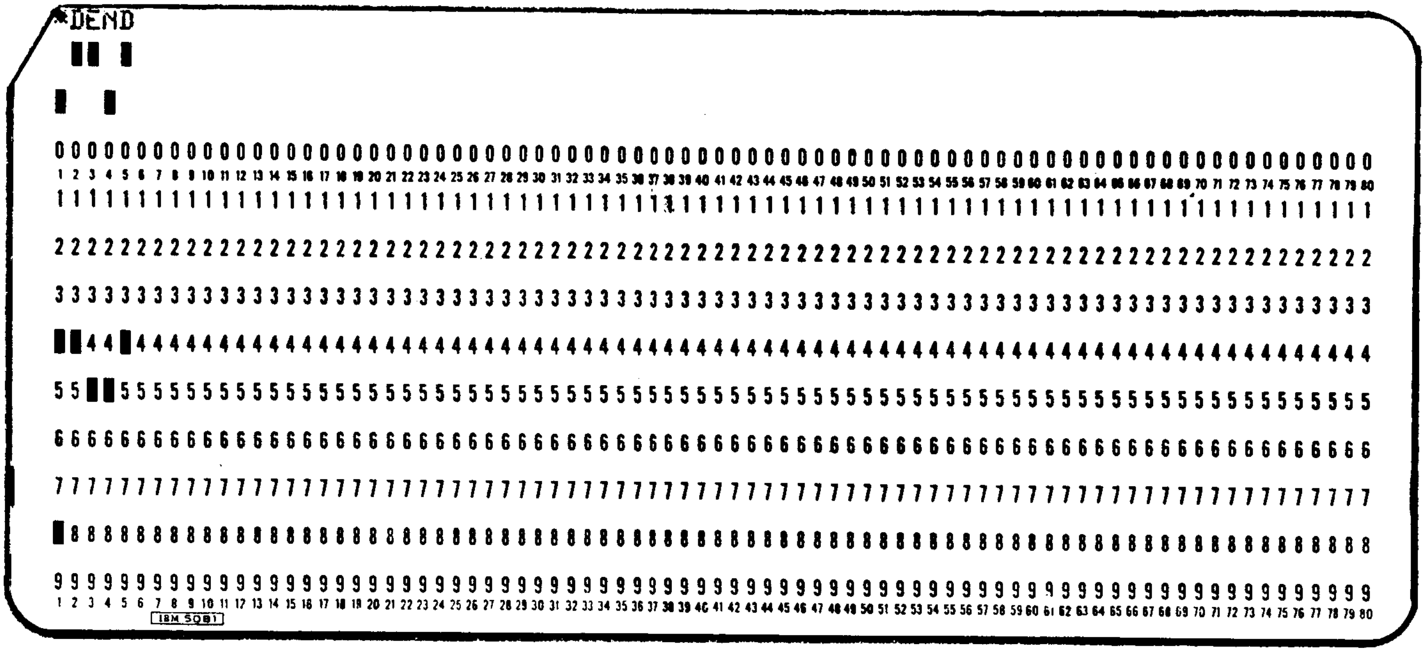


6.2.3.9 Jul/65 **<u>*DEND card</u>** (solid red)

This control card is used by the DEBUGGING PACKAGE and signifies the end of the load-time debugging requests. The *DEND card has the following format:

Column	Contents
1-5	*DEND

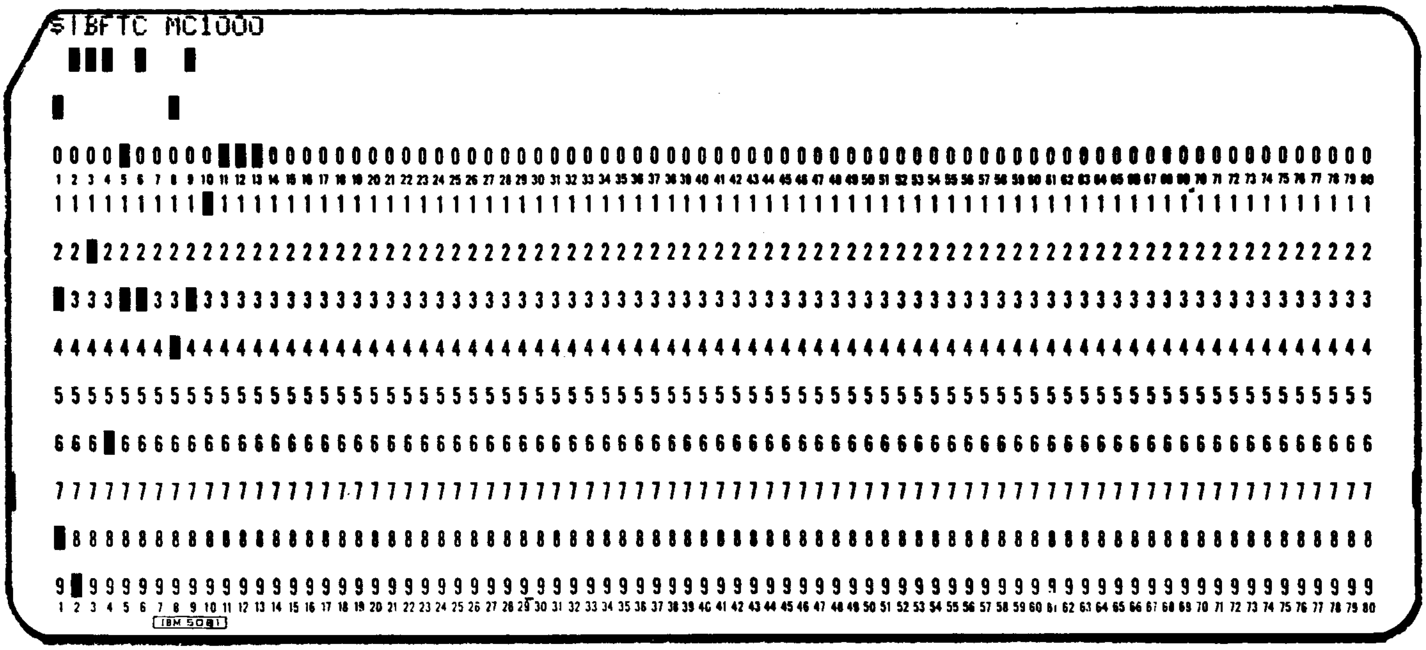
6.2.3.9 (continued)
Jul/65



6.2.3.10 $IBFTC Card (solid orange)
Jul/65

This control card is used by the IBJOB monitor to bring into core the FORTRAN IV compiler. It is necessary that this control card precede each FORTRAN IV source deck within a job. The format of the $IBFTC card is:

Column	Contents
1-6	$IBFTC
8-13	CAB number for deck (must be unique within each job)
16-72	OPTIONS



Notes on Above

The label for the binary deck is obtained from columns 8-13 of this card and may not be omitted. The FORTRAN IV compiler has several output options as well as machine-oriented options that increase the efficiency of the object program. The following options are available to the programmer and may be in any order beginning in column 16, separated by commas:

1) If no options are specified on the $IBFTC card, the compiler:

   (a) produces an object program which uses seven index registers,

   (b) produces a binary deck,

   (c) appends short debug dictionary (SDD) to the binary deck,

   (d) lists the source program, and

   (e) lists all variables used by the source program

2) If the programmer does not wish a binary deck, he may use the following format:

Column	Contents
1-6	$IBFTC
8-13	CAB number for the deck
16-21	NODECK

6.2.3.10 (continued
Jul/65

3) If for some reason the programmer wishes a full debug dictionary for use with the IBM DEBUGGING PACKAGE, he may use the following format:

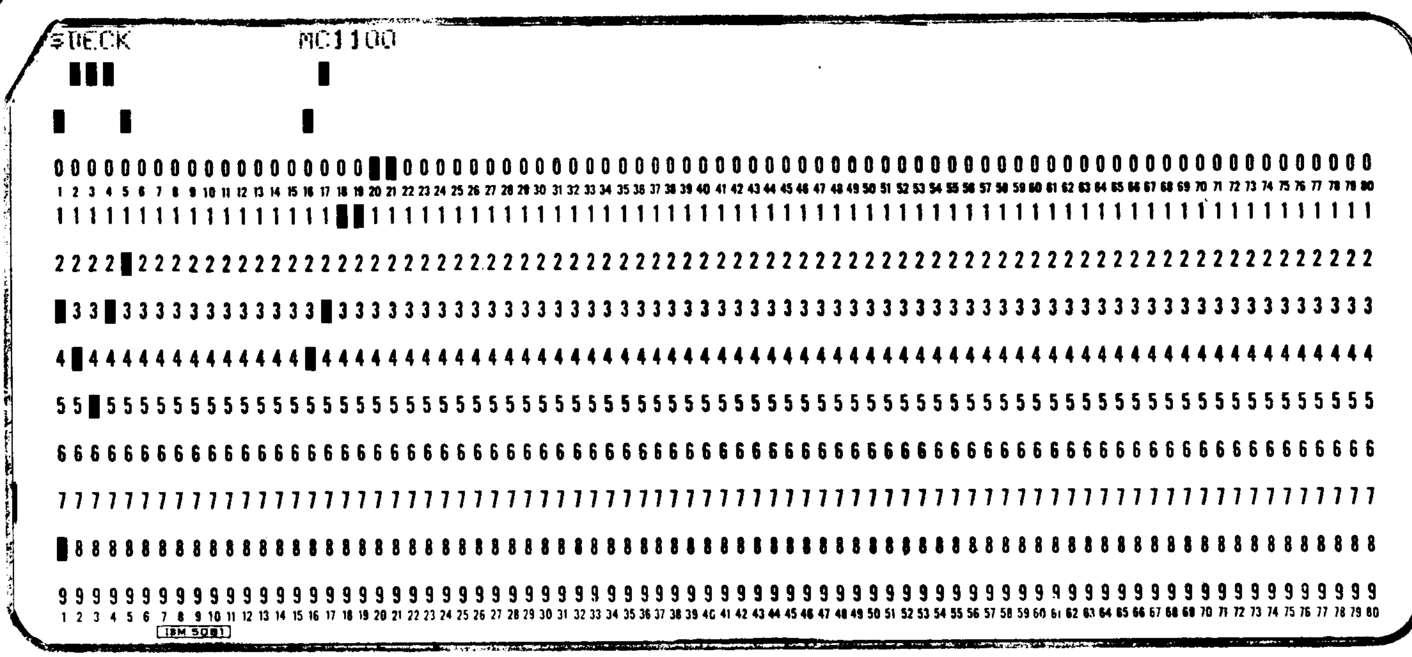
Column	Contents
1-6	$IBFTC
8-13	CAB number for the deck
16-17	DD

Prepunched (no option version) $IBFTC cards are available in the DCS Operations Area.

6.2.3.11 $DECK Card (solid orange)
Nov/65

This control card instructs the 7040 (DCMUP) to obtain the indicated binary deck from the disk and place it in the input stream. There must be a $DECK card for each deck to be loaded from the disk. The $DECK cards precede the $DATA card and have the following format:

Column	Contents
1-5	$DECK
16-21	DECKNAME



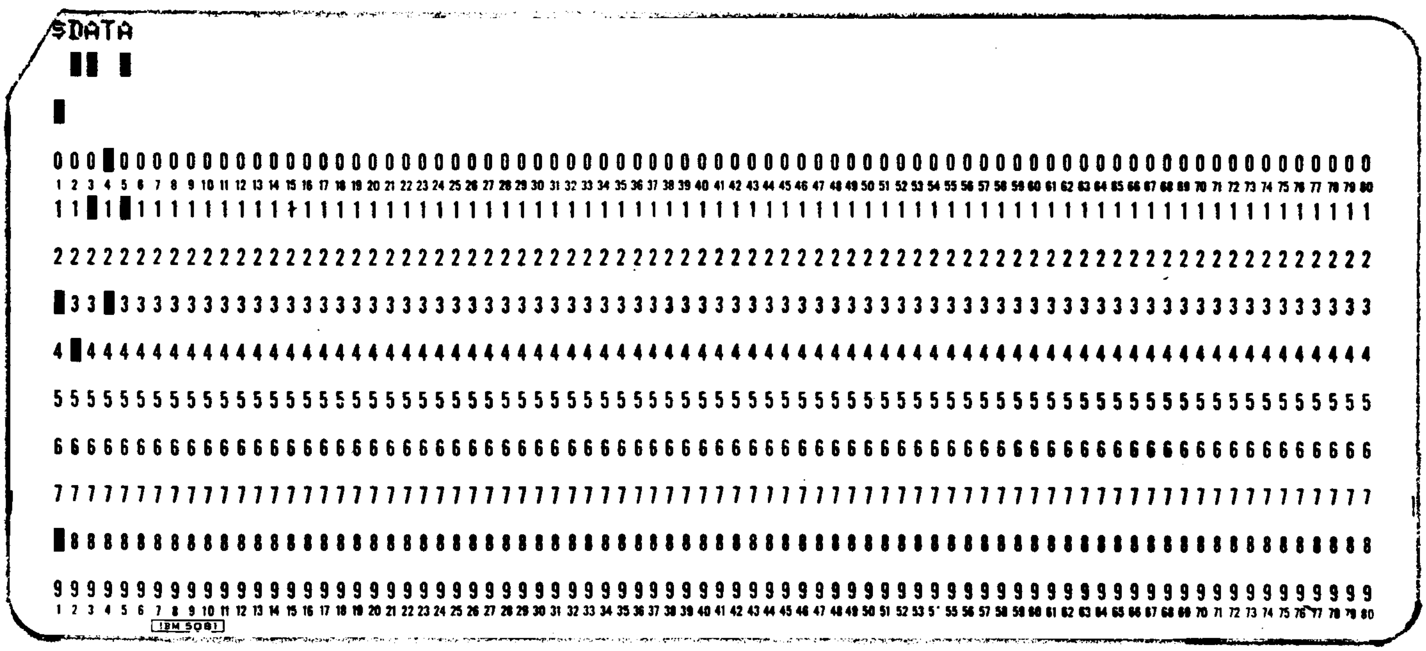
Note on above

DECKNAME Binary decks belong to the FORTRAN IV Card Subroutine Library have decknames AL GAX, AL DERE, AL TRIV, etc. The deckname for any other binary deck is the appropriate CAB number (AB0100, AB0101, MC1100, etc.)

→ 6.2.3.12 Nov/65 **$DATA Card** (solid orange)

This control card precedes the data cards which are pertinent to execution of the program. It is used as an indicator for placement of binary decks and is also used by IBJOB to bring in the loader. It has either of the following formats:

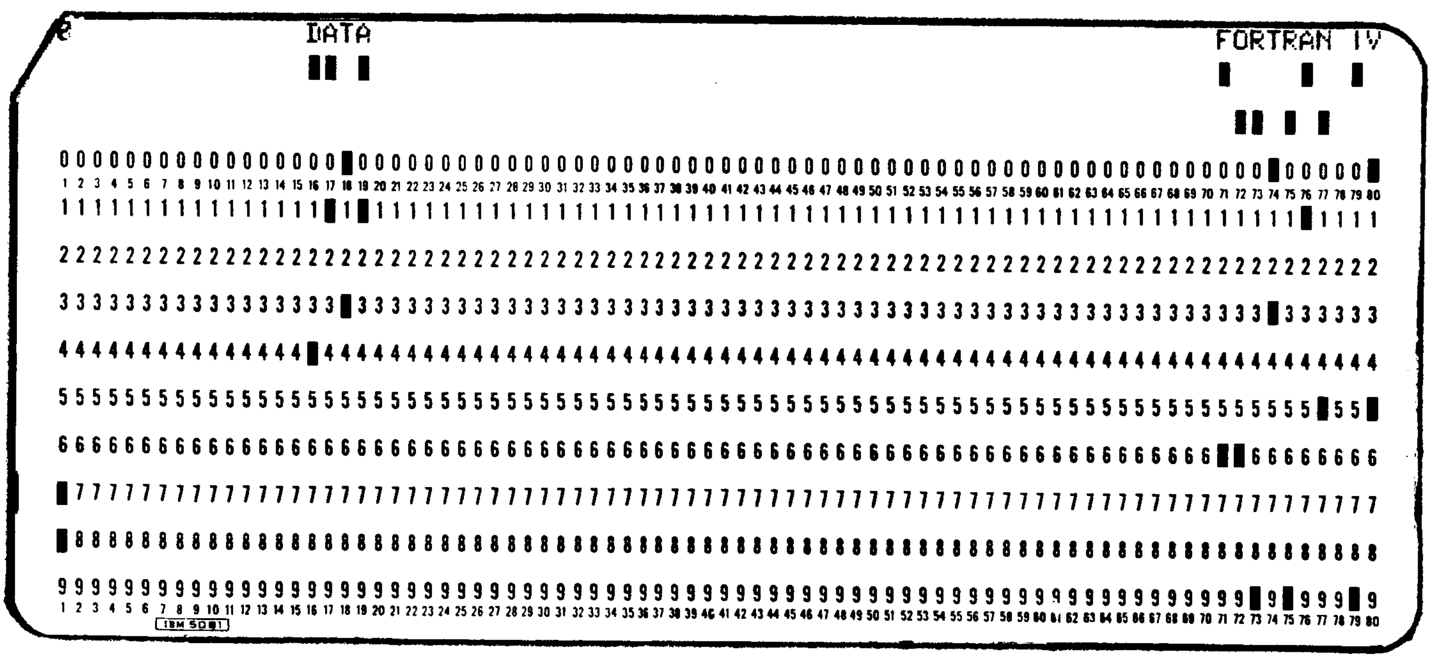
Column	Contents
1-5	$DATA



→ 6.2.3.12 Nov/65 (continued)

or

Column	Contents
1	EOF punch ($^{7}_{8}$)
16-19	DATA
71-80	FORTRAN IV



Prepunched versions of these cards are available at the DCS Operations Area.

→ 6.2.3.13 Nov/65

ADDITIONAL CARDS

There are several other control cards and options available for the IBJOB monitor which have not been described here. They are designed to give more explicit information on the compiler operation and on input/output. These can be used with assistance from members of the CAB staff.

6.3.0 Jul/65 **Debugging**

6.3.1 Jul/65 **Introduction**

In spite of the many advances in programming languages, debugging a program is still one of the most challenging aspects of data processing. It is therefore of extreme importance that the operating system provide the scientist with assistance in this area and that the techniques at his disposal are closely related to the languages with which he has familiarity. There are two times during the operation of the job that debugging aids are necessary. First, when the program is in execution, the user may wish to inspect particular values as they are being computed; secondly, when execution is terminated, the user may want a printout of many of the values which are in memory at that time. In the IBJOB monitor a DEBUGGING PACKAGE has been added. It includes a Debug Compiler (IBDBL) which provides the necessary coding to implement both dynamic dumps (those taken during execution) and post-mortem or static dumps (those taken at the end of execution). In both cases the actual debug requests are supplied in a language which is very similar to FORTRAN IV. This is the first time at Ames that a single debug system has had the capability of producing both types of dumps. This is possible because communication between the Debug Compiler and the IBJOB loader (IBLDR) permits debug requests to be placed in any routine including the system subroutine used for job termination.

In dynamic dumping, the program executes until it reaches the location where the debug request has been placed; the normal program flow is then interrupted, the dump request is executed, and control is returned to the program. Post-mortem dumping occurs in the same manner. However, in this case the debug request has not been inserted in the user's program but rather in a monitor subroutine which controls the terminal activities of a job. When the debug request is finished, program control is transferred to this monitor routine which then completes the final phase of the job and returns control to the IBJOB monitor. Since the processing of debug requests is done in the above manner, it is apparent that additional information must be supplied in both the compilation and execution phases of a job.

Complete details concerning the use of the DEBUGGING PACKAGE are available in the Preface and pages 7-12 of the "IBJOB PROCESSOR DEBUGGING PACKAGE" manual. However, the following sections outline some of the more critical facts and provide additional illustrative examples.

6.3.2 Jul/65 **Compile-Time Requirement**

When a binary deck is produced for a routine it must also contain a debug dictionary, which is the means of communication between the program being debugged and the DEBUGGING PACKAGE. To provide this information the FORTRAN IV compiler automatically produces a short debug dictionary (SDD) for every binary deck. Pertinent data concerning those symbols and statement numbers that appear in the FORTRAN IV source program are contained in this dictionary. If necessary in

6.3.2 Jul/65 (continued)

extreme circumstances, the programmer may obtain a full debug dictionary (DD) by exercising the DD option on the $IBFTC card as described earlier in this manual. In this case all symbols which are generated in the assembled program will appear in the dictionary. Normally this option is not used since (1) more storage is required for the DEBUGGING PACKAGE, (2) compiler time is increased, and (3) the binary deck becomes larger than necessary.

6.3.3 Jul/65 Load Time Requirement

To initiate debugging action for dynamic or post-mortem dumping, the programmer must include a debugging packet. This packet is placed immediately following the $IBJOB card, and its composition is:

1. $IBDBL card, solid red
2. *DEBUG card, solid red
3. *ETC card, solid red (if needed)
4. Debug Request card(s)
5. *DEND

   2, 3, and 4 are repeated for every deck which is to contain a debug request.

The control cards mentioned above are described in the section on control cards in this manual but will be described again for convenience.

1. $IBDBL card (solid red)
   This card instructs the IBJOB monitor to load the DEBUGGING PACKAGE and is also used to specify the amount of debugging output desired. It has the following format:

Column	Contents
1-6	$IBDBL
16-72	OPTIONS Trap Max=n_1, Line Max=n_2 where n_1 and n_2 are integers (with no leading 0) in the range 1-32767. Either or both options in either order may be used to control the amount of debugging output:

Trap Max=n_1 causes debugging action to cease after n_1 requests have been executed. Every debug request encountered, irrespective of any resultant action, is counted.

Line Max=n_2 causes debugging action to cease after approximately n_2 lines of debugging output have been written.

If the programmer does not specify an option he will receive a maximum of 1000 lines of debugging output.

2. <u>*DEBUG card</u> (solid red)
   This card specifies in which deck, and where in that deck, the debug request is to be inserted. It has the following format:

Column	Contents
1-6	*DEBUG
8-13	DECKNAME (CAB number)
16-72	VARIABLE FIELD loc1, loc2, loc3, ...

DECKNAME refers to the object deck in which the debugging action will occur. For a post-mortem dump, the DECKNAME to use is .LXCON (see Post-Mortem example). For a dynamic dump, DECKNAME is the CAB number (AB1000, AB1001, ...) of the deck for which the debug request applies. If this field is blank, the last DECKNAME specified on the preceding *DEBUG card is assumed; if a DECKNAME was not previously specified, the request is deleted.

VARIABLE FIELD, loc1, loc2, loc3, ... refers to the location(s) of the executable statement(s) at which this debug request is to be inserted. For post-mortem dumps the locations to use are: /.LXRTN/, /.LXERR/, /.LXSTR/, /.LXSTP/ (see Post-Mortem example section 6.3.8.1). For dynamic dumps the locations are specified by FORTRAN statement numbers.

3. <u>*ETC card</u> (solid red)
   This card is used when it is necessary to extend the variable field of the *DEBUG card, and as many *ETC cards as are necessary may be used. It has the following format:

Column	Contents
1-4	*ETC
16-72	Extension of the variable field of the *DEBUG card.

4. <u>Debug Request cards</u> (Section 6.3.4)

5. <u>*DEND card</u> (solid red)
   This is always the last card of the debug packet. It informs the DEBUGGING PACKAGE that there are no more debug requests for this job. It has the following format:

Column	Contents
1-5	*DEND

6.3.4
Jul/65

Debug Request Cards

These cards specify the variables or areas of core to be dumped and the frequency of these dumps. In addition, by use of these cards it is possible to (1) introduce, evaluate, and print new variables, (2) assign new values to object deck variables, and (3) print out messages. Debug request statements are written in the FORTRAN IV statement format (a few minor differences exist which are noted later), and the text resembles a FORTRAN IV program.

The following are a few simple examples of a debug request:

Example 1

```
1      78      16
*DEBUG AB1000  10,20

       DUMPbA,B,C,/COM1/
```

The variables A, B, C and all the variables in the common block, /COM1/, are dumped each time statements 10 and 20 in deck AB1000 are encountered.

Example 2

```
1      78      16
*DEBUG AB1000  25

       ON 1,17,3 DUMPbA,B,C,ARRAYX
```

The variables A, B, C and the entire array, ARRAYX, are dumped the 1st, 4th, 7th, 10th, 13th, and 16th times statement 25 in deck AB1000 is encountered.

Example 3

```
1      78      16
*DEBUG AB1000  25

       IF(Ab.LT.bB) DUMPbA,B,C

       ON 1,3 DUMPb/COM1/
```

A is tested each time statement 25 is encountered, and A, B, and C are dumped only if A is less than B. The common block, /COM1/, is dumped the first 3 times statement 25 in deck AB1000 is encountered.

Example 4

```
1      78      16
*DEBUG AB1000  25

       IF(Ab.LT.b0.0) STOP

       IF(Ab.LT.bB) SETbA=B

       ON 1,20,2 DUMPbA,B,C
```

Each time statement 25 in AB1000 is encountered, A is tested. If A is negative, the job is terminated. If A is positive and less than B, then A is assigned the value of B. For all positive values of A the variables A, B, and C are dumped the first time statement 25 is encountered and every other time thereafter until the 20th time.

6.3.4 Jul/65 (continued)

Example 5

```
1      78      16
*DEBUG AB1000  25
       IF(Bb.LE.b15.5) RETURN
       DUMPb'B IS TOO LARGE'
       STOP
```

Each time statement 25 in deck AB1000 is encountered, B is tested. If it is greater than 15.5, the message, 'B IS TOO LARGE', is printed and the job is terminated. Otherwise, execution of the object program continues.

6.3.5 Jul/65 Debug Request Statements

6.3.5.1 Jul/65 Table of Debug Request Statements

Statement	Usage
1. STOP	Controls flow of debugging action.
2. CALL	
3. RETURN	
4. Unconditional GO TO	
5. Logical IF	
6. ON	Controls the time when debugging action is to occur.
7. SET	By use of an arithmetic statement, object program variables or debug packet variables can be set to new values.
8. NAME	Permits the programmer to introduce and define debug packet variables.
9. DUMP	Causes dumping of the specified variables or regions of core.
10. LIST	Specifies a list of variables or regions of core which can be used by the DUMP statement.
11. COMMENT CARD	Comments.

6.3.5.2 Jul/65

Brief Description of Debug Request Statements

1. STOP

   STOP causes the execution of a job to cease and the control to be transferred to EXIT (.LXCON).

2. CALL

   CALLbSUB(ARG1,ARG2) — within a debug request the programmer may call any of the object deck subroutines or library tape subroutines.

3. RETURN

   RETURN is used when the programmer wishes to leave the debug packet to continue with his object program. There is an implicit RETURN at the end of each debug request.

4. Unconditional GO TO

   This causes a transfer within the debug packet. GO TO 5 transfers control to statement 5 of the debug packet.

5. Logical IF

   The form and usage of the debugging logical IF is similar to that of FORTRAN IV.

   Examples: IFbALPHAb.GT.b2.0b DUMPbALPHA, BETA

   IF(BETAbGTb10.0) RETURN

   IFbALPHA*BETAbLTb0.01b GO TO 5

   IFbAb.GE.bBbANDbCbLTbDb STOP

   Notice that a blank must always precede and follow the logical and relational operators and that the decimal point is optional (b.GT.b is same as bGTb).

6. NAME

   NAME is used to introduce and define new variables for debug request usage. The NAME statement for these variables must appear before any reference to them is made. The mode of the variable must be specified and can be any of the following:

   Ø Octal
   F Floating point number
   X Integer
   D Double precision floating point number
   J Complex number
   L Logical
   H Alphanumeric

   A slash must immediately follow a variable's name.

   Examples: 1) NAME XTAB/=NEW(F(50))
   The new variable XTAB is defined as a floating point array whose size is 50. In the above (F(50)) cannot be written (Fb(50)).

2) NAME CNTR/=NEW(X)
The new variable CNTR is defined as an integer.

7. ON

ON is used to control the frequency of the debugging action.

Examples: 1)

```
1      8          16
*DEBUG EF1000     35
       7
       ON   1,22,5     DUMPbR
```

R will be dumped on the 1st, 6th, 11th, 16th, and 21st times statement 35 is encountered.

2)

```
1      8          16
*DEBUG EF1000     35
       7
       ON 1,22,5 IF(RbGTbU) DUMPbR,U
```

R and U will be dumped the 1st, 6th, 11th, 16th, and 21st times statement 35 is encountered only if R is greater than U.

3)

```
1      8          16
*DEBUG EF1000     35
       7
       ON(COUNT) 1,15,3  DUMPbR
```

The debug packet symbol COUNT is created and defined as an integer. R will be dumped when COUNT is 1,4,7,10 and 13. Any statement following this ON statement may reference COUNT in any way. If, however, reference to COUNT is necessary prior to the ON statement, then the programmer must first introduce and define COUNT as an integer using the NAME statement:

NAME COUNT/=NEW(X)

8. SET

SET is used to give a new value to an object program variable or debug packet variable.

```
SETbCONTR=0
SETbA=1.4
IF(Xb.LT.bY)  SETbX=Y
SETbB=(C+D)/2.0
```

The arithmetic statements used with SET must not contain functions (SIN, SQRT, etc.), **, nor the logical constants .TRUE. and .FALSE.

9. DUMP

The following examples illustrate how to dump variables and regions of core and how to print messages.

Examples: 1) <u>any single item</u>

a) a variable, DUMPbALPHA
b) an array element, DUMPbFY(6),
If FY is double precision, then locations FY + 10 and FY + 11 are dumped.

2) Any region within the deck being debugged

   a) DUMPb(FIRST,LAST). If FIRST is the first physically mentioned variable in the deck and LAST is the last, then all variables the program uses except those in COMMON will be dumped in their correct modes.
   b) DUMPbARRAYA, (ARRAYB(1),ARRAYB(10)). This will dump all of ARRAYA and the first 10 elements of ARRAYB.
   c) DUMPb(15,25,Ø). This will dump in the octal format that region of the source program which begins with statement 15 and ends with statement 25.

3) Labeled Common

   a) DUMPb/COM1/,/COM2/. This causes the labeled common blocks, COM1 and COM2, to be dumped in their entirety.
   b) DUMPb/COM1/b±n. This will dump the contents of the location which is ±n cells away from the beginning of the common block, COM1.

4) Qualified variables or regions

   In order to dump variables or regions of core, it is not necessary that they belong to the deck being debugged. They may be in any deck provided they are preceded by the appropriate DECKNAME and $$. If more than one variable are being dumped, they must be enclosed in brackets.

   Example:
   DUMPbEF1000$$A,EF1001$$(C,D,E),EF1002$$((FIRST,LAST))
   causes dumping of A in deck EF1000; C, D, and E in deck EF1001; and all the variables, exclusive of COMMON, in EF1002 (provided FIRST is the first physically mentioned variable in the program, and LAST is the last mentioned variable).

5) Variables or regions of core may be dumped by referring to the number of a LIST statement appearing in the debugging packet, i. e.,

   DUMPb29

   where:

   ```
   +  7
   29 LIST A,B,C,(FIRST,LAST),/COM1/,EF1010$$(C,D,E,F)
   ```

   This is useful when the same list occurs more than once within the debugging packet.

   Example: IF(Ab.LE.bB) DUMPb29
   ON 1,20,3 DUMPb29

6) Messages

   a) DUMPb'FX DOES NOT CONVERGE, GO TO NEXT CASE.'
   Quotation marks ($^{8}_{4}$ punch) are used to enclose the message and should not be used in the message itself. The - (dash) on the keypunch is an $^{8}_{4}$ punch.

6.3.6 Jul/65

Debug Request and FORTRAN Statement Differences

While debug requests have a statement format similar to that of FORTRAN IV, there are some differences and restrictions. The following list indicates these.

1. DUMPbX
   CALLbSUB(ARG1,ARG2)   } The blank is required.
2. ** is not permitted.
3. Function references (SIN(X), EXP(X), etc.) are not allowed.
4. Dummy variables cannot be referenced.
5. A decimal integer cannot have a leading zero
6. Subscripts may be any arithmetic expression and will be converted to an integer if necessary.
7. Blanks must be used to separate (1) logical expressions from logical operators and (2) variables from relational operators.

   Xb.OR.bY , Ab.LT.bB .
8. Imbedded blanks are not allowed,

   ALPHA cannot be written ALbPHA

6.3.7 Jul/65

Notes on the Debugging Package

1. The short debugging dictionary (SDD) which is necessary when using the DEBUGGING PACKAGE is supplied automatically.
2. The debugging packet is placed immediately following the $IBJOB card.
3. 1000 lines of debugging output are printed unless more are specified on the $IBDBL card.
4. The estimated lines of output which are specified on the $JOB card should also include the amount of debugging output expected.
5. The TRAP MAX option on the $IBDBL card can be used to stop debugging action.
6. The frequency of debugging can be controlled by the ON statement or by a logical IF statement.
7. The statement number and DECKNAME for the debug request are specified on the *DEBUG card.
8. Debug requests can be placed on any executable statement.
9. Debugging action occurs before execution of the statement itself.
10. Any variable, regardless of mode, may be dumped with the exception of dummy variables.

6.3.7 (continued)
Jul/65

11. Within a single deck it is possible to dump all variables from all other decks provided the variables are qualified by preceding them with a DECKNAME and $$; i.e., EF4700$$(A,B,C,D). It is not necessary to qualify labeled common blocks.

12. Debug requests can be inserted at the entry point of a subroutine. This allows debugging action to occur each time the subroutine is entered and before any computing is performed by the subroutine. The entry point is specified by enclosing the subroutine name within slashes.

```
1         8           16
*DEBUG    EF4008       /SHOCK/, 10, 20
```

13. Dump can be used to print a message by enclosing the message in quotation marks ($^{8}_{4}$ punch). The - (dash) on the keypunch is an $^{8}_{4}$ punch.

    DUMPb'THIS IS A MESSAGE.'

14. A decimal integer cannot have a leading zero.

15. When referencing /COM1/b+3 the blank must be included.

16. Dumping of blank common does not work correctly.

17. Earlier versions of the FORTRAN IV Compiler did not append a debug dictionary to the binary decks it produced. If by mistake the programmer should request debugging action using these decks, the job will execute, and the debug request will be deleted. A LEVEL=2 diagnostic concerning the debug request will be printed.

6.3.8.0 Examples
Jul/65

6.3.8.1 Post Mortem Dump Example
Jul/65

All jobs which operate under the IBJOB monitor automatically enter a control subroutine when they are terminated. The DECKNAME for this control subroutine is .LXCON. Depending on the nature of termination, this subroutine is entered at one of the following entry points:

1. /.LXRTN/ - when a job is terminated by

    a) CALL EXIT
    b) STOP
    c) CALL DUMP
    d) No more input data
    e) Time and line count kickoff

2. /.LXERR/ - when a job is terminated by EXEM, i.e., an input/output error exists.

(continued)

3. /.LXSTR/ } When program logic has been destroyed and erroneous
4. /.LXSTP/ } instructions are being executed.

Since the *DEBUG card specifies which deck and where in that deck debugging action is to occur, the *DEBUG card for a post-mortem dump request has the following format:

```
1     6  8   13    16
*DEBUG  .LXCON     /.LXRTN/,/.LXERR/,/.LXSTR/,/.LXSTP/
```

The variables or regions of core the programmer wishes to have dumped are specified on DUMP cards and must be correctly qualified, i.e.,

```
DUMP  AB0100$$((FIRST,LAST)),/COM1/,/COM2/
```

The following is a typical example of the debugging packet which must be supplied by the user when he desires a post-mortem dump. This debugging packet follows the $IBJOB card and is punched on solid red cards.

```
1    6         16
$IBDBL          LINEbMAX=1500
1    6 8       16
*DEBUGb.LXCON   /.LXERR/,/.LXSTR/,/.LXSTP/
*ETC             ,/.LXRTN/                    (See Note 2.)
      7
      DUMPbAB0100$$((FIRST,LAST)),/COM1/,/COM2/
      DUMPbAB0101$$((FIRST,LAST))
      DUMPbAB0102$$((FIRST,LAST))
      DUMPbAB0103$$((FIRST,LAST)),/COM3/,/COM4/
      DUMPbAB0104$$((FIRST,LAST))
      DUMPbAB0115$$((FIRST,LAST))
*DEND
```

Explanation: Approximately 1500 lines of debugging output were expected and therefore requested on the $IBDBL card. If no request is made, the programmer will receive a maximum of 1000 lines.

All variables, except common variables, in all decks will be dumped. In addition labeled common blocks, COM1 and COM2, will be dumped from deck AB0100, and common blocks, COM3 and COM4, will be dumped from deck AB0103. The symbol FIRST refers to the first physically mentioned variable within a deck, and LAST refers to the last mentioned variable. They can have any name. A single DUMP statement with continuation cards could replace the above 6 DUMP statements.

Note 1.
The programmer may, if he wishes, use the DUMP statement in other ways for post-mortem dumps (see description of DUMP in statement 6.3.5.2).

Note 2.
When a job is in production, it is desirable to receive a post-mortem dump only if the job has <u>not</u> entered .LXCON at entry point /.LXRTN/. By placing /.LXRTN/ on an *ETC card, the debugging packet is easily

6.3.8.1 (continued)
Nov/65

altered for production runs. With the *ETC card removed the debugging packet will give a dump only if the job has an input/output error or if the job is executing erroneous instructions.

Note 3.

POST-MORTEM DUMPS, ALCRASH -
Labeled post-mortem dumps may also be obtained by using the subroutine ALCRASH. Each deck for which a post-mortem dump is desired must contain a CALL CRASH (N,VLIST, A, B, I, D, ...) statement. It must also have a DATA statement associated with it to give the names, modes, and dimensions of the variables to be dumped. Complete details for usage are contained in the write-up of AL CRASH in the Ames FORTRAN IV Subroutine Library manual. CRASH will have the following effect when it is being used for post-mortem dumps. At the end of the execution phase a labeled dump will be printed of all variables listed in each CRASH statement and in the order the CRASH statements were encountered. This also gives a condensed flow trace of the execution of the program. This post-mortem dump will be given on DEBUG or RECOM jobs regardless of how execution was terminated. In PRODb jobs the post-mortem dump will be given only if the job comes to an unexpected stop or goes into a tight loop and is manually stopped.

→ If the programmer wishes to use the subroutine ALCRASH he must supply the following $DECK card:

^{1}\$DECK ^{16}ALCRSH

a dummy CRASH routine exists on the library tape.

6.3.8.2 Dynamic Dump Examples
Jul/65

```
$IBDBL          TRAP MAX = 450, LINE MAX = 500
*DEBUG MAIN     25
      NAME A/=NEW(F)
      ON 1  SET A=1.4
      IF BETA GE A RETURN
      DUMP BETA, 'BETA TOO SMALL.'
      SET A=A-0.1
*DEBUG SUBR     1
      NAME X/=NEW(X)
      SET X=0
*DEBUG SUBR     30
      ON(X) 1,Z,3 DUMP MAIN$$ARRAYA
*DEND
```

```
CMAIN   M.PERNICIARO     CHECK OUT OF DEBUGGING EXAMPLE IN DEBUG MANUAL.
      DIMENSION ARRAYA(5)
      BETA=1.2                                        ,1
      DO 25 I=1,3                                     ,2     ,3
   25 CONTINUE                                        ,4     ,5
      DO 26 I=1,5                                     ,6     ,7
      K=I                                             ,8
      CALL SUBR(ARRAYA,K)                             ,9
   26 CONTINUE                                        ,10    ,11
      STOP                                            ,12
      END                                             ,13
```

```
CSUBR   M.PERNICIARO     SUBROUTINE FOR CHECK OUT OF DEBUGGING EXAMPLE.
      SUBROUTINE SUBR(D,K)
      DIMENSION D(5)
      Z=15                                            ,1
    1 DO 30 I=1,2                                     ,2     ,3
      D(K)=K+I*10                                     ,4
   30 CONTINUE                                        ,5     ,6
      RETURN                                          ,7
      END                                             ,8
```

LISTING OF DEBUGGING DUMPS

```
1,   ******** DUMP REQUEST AT 25., REL LOC 00020, ABS LOC 03000, IN DECK MAIN

                    (BETA,BETA,F)

 MAIN         BETA        ......+41
 00051 03031 BETA         +.12000000+01
 BETA TOO SMALL.

2,   ******** DUMP REQUEST AT 25., REL LOC 00020, ABS LOC 03000, IN DECK MAIN

                    (BETA,BETA,F)

 MAIN         BETA        ......+41
 00051 03031 BETA         +.12000000+01
 BETA TOO SMALL.

3,   ******** DUMP REQUEST AT 30., REL LOC 00054, ABS LOC 03113, IN DECK SUBR

                    (ARRAYA,ARRAYA+4,F)

 MAIN         ARRAYA      ......+36      (5)
 00044 03024 ARRAYA       +.11000000+02  -.00000000+00  -.00000000+00  -.00000000+00  -.00000000+00

4,   ******** DUMP REQUEST AT 30., REL LOC 00054, ABS LOC 03113, IN DECK SUBR

                    (ARRAYA,ARRAYA+4,F)

 MAIN         ARRAYA      ......+36      (5)
 00044 03024 ARRAYA       +.21000000+02  +.12000000+02  -.00000000+00  -.00000000+00  -.00000000+00
```

```
5,    ******** DUMP REQUEST AT 30., REL LOC 00054, ABS LOC 03113, IN DECK SUBR

                    (ARRAYA,ARRAYA+4,F)

 MAIN         ARRAYA      ......+36        (5)
 00044 03024  ARRAYA      +.21000000+02    +.22000000+02    +.13000000+02    -.00000000+00    -.00000000+00

6,    ******** DUMP REQUEST AT 30., REL LOC 00054, ABS LOC 03113, IN DECK SUBR

                    (ARRAYA,ARRAYA+4,F)

 MAIN         ARRAYA      ......+36        (5)
 00044 03024  ARRAYA      +.21000000+02    +.22000000+02    +.23000000+02    +.14000000+02    -.00000000+00

7,    ******** DUMP REQUEST AT 30., REL LOC 00054, ABS LOC 03113, IN DECK SUBR

                    (ARRAYA,ARRAYA+4,F)

 MAIN         ARRAYA      ......+36        (5)
 00044 03024  ARRAYA      +.21000000+02    +.22000000+02    +.23000000+02    +.24000000+02    +.15000000+02

          214 LINES OUTPUT.
```

```
$IBDBL
*DEBUG PF2020  19
      ON 1,40,6 DUMP A,B,C
      ON 1,1 DUMP /COM1/
*DEBUG PF2020  20
      ON 1,3 DUMP A,B,C
*DEBUG PF2021  30
      IF(DISC .LT. 0.0) ON 1,40 DUMP DISC,A,B,C
      ON 1,40,5 DUMP DISC
      ON 1,40,6 DUMP (/COM1/,/COM1/ +2)
*DEBUG PF2021  71
      ON 1,40,7 DUMP S
*DEND
```

```
C        PS2020
      COMMON /COM1/A,B,C,X1R,X1I,X2R,X2I
   21 NPAGE=1                                                      ,1
    5 WRITE (6,8) NPAGE                                            ,2     ,3     ,4
    8 FORMAT(117H1         A               B               C              X1 REAL
     1         X1 IMAG        X2 REAL        X2 IMAG               PAGE 14//)
      NPAGE=NPAGE+1                                                ,5
      LINES=0                                                      ,6
   14 READ (5,15) A,B,C                                            ,7     ,8     ,9
   15 FORMAT(3F10.5)
   20 IF (A) 16,17,16                                              ,10
   17 WRITE (6,18)                                                 ,11    ,12
   18 FORMAT(13H JOB FINISHED)
      GO TO 14                                                     ,13
   16 CALL SOLVE                                                   ,14
      IF (X1I) 90,91,90                                            ,15
   91 WRITE (6,95) A,B,C,X1R,X2R                                   ,16    ,17    ,18
   95 FORMAT(1H ,1P4E14.4,E28.4)
      GO TO 19                                                     ,19
   90 IF (X1R) 100,101,100                                         ,20
  101 IF (X2R) 100,102,100                                         ,21
  102 WRITE (6,103) A,B,C,X1I,X2I                                  ,22    ,23    ,24
  103 FORMAT(1H ,1P3E14.4,2E28.4)
      GO TO 19                                                     ,25
  100 WRITE (6,110) A,B,C,X1R,X1I,X2R,X2I                          ,26    ,27    ,28
  110 FORMAT(1H ,1P7E14.4)
   19 LINES=LINES+1                                                ,29
      IF (LINES.GE.40) GO TO 5                                     ,30    ,31    ,32
      GO TO 14                                                     ,33
      END                                                          ,34
```

```
     #62021
      SUBROUTINE SOLVE
      COMMON /COM1/A,B,C,X1R,X1I,X2R,X2I
      DISC=B**2-4.0*A*C                                    ,1
   30 DUM=DUM                                              ,2
      IF (DISC) 50,60,70                                   ,3
   50 X1R=-B/(2.0*A)                                       ,4
      X2R=X1R                                              ,5
      X1I=SQRT (-DISC)/(2.0*A)                             ,6
      X2I=-X1I                                             ,7
      RETURN                                               ,8
   60 X1R=-B/(2.0*A)                                       ,9
      X2R=X1R                                              ,10
      X1I=0.0                                              ,11
      X2I=0.0                                              ,12
      RETURN                                               ,13
   70 S=SQRT(DISC)                                         ,14
   71 A=A                                                  ,15
      X1R=(-B+S)/(2.0*A)                                   ,16
      X2R=(-B-S)/(2.0*A)                                   ,17
      X1I=0.0                                              ,18
      X2I=0.0                                              ,19
      RETURN                                               ,20
      END                                                  ,21
```

A	B	C	X1 REAL	X1 IMAG	X2 REAL	X2 IMAG
(BUFFER AT 17403)						
DUMP NUMBER 00001						
DUMP NUMBER 00002						
DUMP NUMBER 00003						
4.0000E 00	-3.0000E 00	-5.0000E 00	1.5542E 00		-8.0425E-01	
DUMP NUMBER 00004						
DUMP NUMBER 00005						
1.5000E 01	-1.1000E 01	-1.4000E 01	1.4000E 00		-6.6667E-01	
DUMP NUMBER 00006						
5.0000E 00	-4.0000E 00	-3.0000E 00	1.2718E 00		-4.7178E-01	
DUMP NUMBER 00007						
3.0000E 00	-2.0000E 00	4.0000E 00	3.3333E-01	1.1055E 00	3.3333E-01	-1.1055E 00
8.0000E 00	5.0000E 00	0.	-0.		-6.2500E-01	
DUMP NUMBER 00008						
1.0000E 00	9.0000E 00	1.0000E 00	-1.1252E-01		-8.8875E 00	
DUMP NUMBER 00009						
4.0000E 00	5.0000E-01	1.0000E-01	-6.2500E-02	1.4524E-01	-6.2500E-02	-1.4524E-01
DUMP NUMBER 00010						
DUMP NUMBER 00011						
-3.0000E 00	5.0000E 00	-6.0000E 00	8.3333E-01	-1.1426E 00	8.3333E-01	1.1426E 00
DUMP NUMBER 00012						
9.0000E 00	-1.2000E 01	4.0000E 00	6.6667E-01		6.6667E-01	
1.0000E 00	-5.0000E 00	6.0000E 00	3.0000E 00		2.0000E 00	
DUMP NUMBER 00013						
3.0000E 00	-5.0000E 00	7.0000E 00	8.3333E-01	1.2802E 00	8.3333E-01	-1.2802E 00
DUMP NUMBER 00014						
5.0000E 00	-3.0000E 00	2.0000E 00	3.0000E-01	5.5678E-01	3.0000E-01	-5.5678E-01
DUMP NUMBER 00015						
6.0000E 00	-2.0000E 00	3.0000E 00	1.6667E-01	6.8718E-01	1.6667E-01	-6.8718E-01
DUMP NUMBER 00016						
8.0000E 00	0.	-7.0000E 00	9.3541E-01		-9.3541E-01	
DUMP NUMBER 00017						
9.0000E 00	2.5000E 01	-3.0000E 01	9.0509E-01		-3.6829E 00	
DUMP NUMBER 00018						
4.0000E 00	-2.0000E 01	2.5000E 01	2.5000E 00		2.5000E 00	
DUMP NUMBER 00019						
5.0000E 00	-2.0000E 00	1.0000E 00	2.0000E-01	4.0000E-01	2.0000E-01	-4.0000E-01
5.0000E 00	-3.0000E 00	0.	6.0000E-01		0.	

LISTING OF DEBUGGING DUMPS

```
1.   ******** DUMP REQUEST AT 20., REL LOC 00043, ABS LOC 03047, IN DECK PF2020

                    (A,A,F)

 PF2020        A            COM1
 00235 03242 A              +.14000000+01

                    (B,B,F)

 PF2020        B            COM1   +1
 00236 03243 B              -.30000000+01

                    (C,C,F)

 PF2020        C            COM1   +2
 00237 03244 C              -.50000000+01


2.   ******** DUMP REQUEST AT 30., REL LOC 00022, ABS LOC 03300, IN DECK PF2021

                    (DISC,DISC,F)

 PF2021        DISC         SOLVE +88
 00130 03406 DISC           +.18900000+02

                    (A,C)

 PF2020        A            COM1
 00235 03242 A              +.14000000+01
 00236 03243 B              -.30000000+01
 00237 03244 C              -.50000000+01


3.   ******** DUMP REQUEST AT 71., REL LOC 00101, ABS LOC 03357, IN DECK PF2021

                    (S,S,F)
```

```
PF2021       S          SOLVE +90
00132 03410  S          +.94339811+01


4.   ******** DUMP REQUEST AT 19.., REL LOC 00153, ABS LOC 03157, IN DECK PF2020

                  (A,A,F)

PF2020       A          COM1
00235 03242  A          +.14000000+01

                  (B,B,F)

PF2020       B          COM1   +1
00236 03243  B          -.30000000+01

                  (C,C,F)

PF2020       C          COM1   +2
00237 03244  C          -.50000000+01

                  (A,X2I)

PF2020       A          COM1
00235 03242  A          +.14000000+01
00236 03243  B          -.30000000+01
00237 03244  C          -.50000000+01
00240 03245  K1R        +.15542476+01
00241 03246  K1I        +.00000000+00
00242 03247  K2R        -.80424764+00
00243 03250  K2I        +.00000000+00


5.   ******** DUMP REQUEST AT 20.., REL LOC 00043, ABS LOC 03047, IN DECK PF2020

                  (A,A,F)

PF2020       A          COM1
00235 03242  A          +.15000000+02
```

```
                    (B,B,F)

PF2020       B            COM1   +1
00236 03243  B            -.11000000+02

                    (C,C,F)

PF2020       C            COM1   +2
00237 03244  C            -.14000000+02


6.   ******** DUMP REQUEST AT 20., REL LOC 00043, ABS LOC 030476 IN DECK PF2020

                    (A,A,F)

PF2020       A            COM1
00235 03242  A            +.15000000+01

                    (B,B,F)

PF2020       B            COM1   +1
00236 03243  B            -.40000000+01

                    (C,C,F)

PF2020       C            COM1   +2
00237 03244  C            -.30000000+01


7.   ******** DUMP REQUEST AT 30., REL LOC 00022, ABS LOC 033006 IN DECK PF2021

                    (DISC,DISC,F)

PF2021       DISC         SOLVE +88
00130 03406  DISC         -.44000000+02

                    (A,A,F)

PF2020       A            COM1
00235 03242  A            +.30000000+01
```

```
                  (B,B,F)

PF2020       B            COM1  +1
00236 03243  B            -.20000000+01

                  (C,C,F)

PF2020       C            COM1  +2
00237 03244  C            +.14000000+01


8.   ******** DUMP REQUEST AT 30., REL LOC 00022, ABS LOC 03300, IN DECK PF2021

                  (DISC,DISC,F)

PF2021       DISC         SOLVE +88
00130 03406  DISC         +.17700000+02


9.   ******** DUMP REQUEST AT 30., REL LOC 00022, ABS LOC 03300, IN DECK PF2021

                  (DISC,DISC,F)

PF2021       DISC         SOLVE +88
00130 03406  DISC         -.13500000+01

                  (A,A,F)

PF2020       A            COM1
00235 03242  A            +.40000000+01

                  (B,B,F)

PF2020       B            COM1  +1
00236 03243  B            +.15000000+00

                  (C,C,F)

PF2020       C            COM1  +2
00237 03244  C            +.19999999-01
```

```
                    (A,C)

PF2020        A           COM1
00235 03242 A             +.40000000+01
00236 03243 B             +.50000000+00
00237 03244 C             +.99999999-01


10.  ******** DUMP REQUEST AT 19., REL LOC 00153, ABS LOC 03157, IN DECK PF2020

                    (A,A,F)

PF2020        A           COM1
00235 03242 A             +.40000000+01

                    (B,B,F)

PF2020        B           COM1   +1
00236 03243 B             +.50000000+00

                    (C,C,F)

PF2020        C           COM1   +2
00237 03244 C             +.99999999-01


11.  ******** DUMP REQUEST AT 30., REL LOC 00022, ABS LOC 03300, IN DECK PF2021

                    (DISC,DISC,F)

PF2021        DISC        SOLVE +88
00130 03406 DISC          -.47000000+02

                    (A,A,F)

PF2020        A           COM1
00235 03242 A             -.30000000+01

                    (B,B,F)
```

```
PF2020       B          COM1  +1
00236 03243 B          +.50000000+01

                 (C,C,F)

PF2020       C          COM1  +2
00237 03244 C          -.60000000+01


12.  ******** DUMP REQUEST AT 30., REL LOC 00022, ABS LOC 03300, IN DECK PF2021

                 (DISC,DISC,F)

PF2021       DISC       SOLVE +88
00130 03406 DISC       +.10000000+00

                 (A,A,F)

PF2020       A          COM1
00235 03242 A          +.90000000+01

                 (B,B,F)

PF2020       B          COM1  +1
00236 03243 B          -.12000000+02

                 (C,C,F)

PF2020       C          COM1  +2
00237 03244 C          +.14000000+01


13.  ******** DUMP REQUEST AT 30., REL LOC 00022, ABS LOC 03300, IN DECK PF2021

                 (DISC,DISC,F)

PF2021       DISC       SOLVE +88
00130 03406 DISC       -.59000000+02

                 (A,A,F)
```

```
PF2020        A            COM1
00235 03242 A            +.30000000+01

                (B,B,F)

PF2020        B            COM1   +1
00236 03243 B            -.50000000+01

                (C,C,F)

PF2020        C            COM1   +2
00237 03244 C            +.70000000+01

                (DISC,DISC,F)

PF2021        DISC         SØLVE +88
00130 03406 DISC         -.59000000+02

14.  ******** DUMP REQUEST AT 30., REL LOC 00022, ABS LØC 033006 IN DECK PF2021

                (DISC,DISC,F)

PF2021        DISC         SØLVE +88
00130 03406 DISC         -.31000000+02

                (A,A,F)

PF2020        A            COM1
00235 03242 A            +.50000000+01

                (B,B,F)

PF2020        B            COM1   +1
00236 03243 B            -.30000000+01

                (C,C,F)

PF2020        C            COM1   +2
00237 03244 C            +.20000000+01
```

```
15.  ******** DUMP REQUEST AT 30.. REL LOC 00022, ABS LOC 03300, IN DECK PF2021

                 (DISC,DISC,F)

PF2021        DISC        SOLVE +88
00130 03406  DISC        -.68000000+02

                 (A,A,F)

PF2020        A           COM1
00235 03242  A           +.60000000+01

                 (B,B,F)

PF2020        B           COM1   +1
00236 03243  B           -.20000000+01

                 (C,C,F)

PF2020        C           COM1   +2
00237 03244  C           +.30000000+01

                 (A,C)

PF2020        A           COM1
00235 03242  A           +.60000000+01
00236 03243  B           -.20000000+01
00237 03244  C           +.30000000+01


16.  ******** DUMP REQUEST AT 19.. RBL LOC 00153, ABS LOC 03157, IN DECK PF2020

                 (A,A,F)

PF2020        A           COM1
00235 03242  A           +.60000000+01

                 (B,B,F)
```

```
PF2020       B            COM1  +1
00236 03243 B            -.20000000+01

              (C,C,F)

PF2020       C            COM1  +2
00237 03244 C            +.30000000+01


17.  ******** DUMP REQUEST AT 71., REL LOC 00101, ABS LOC 03357, IN DECK PF2021

              (S,S,F)

PF2021       S            SOLVE +90
00132 03410 S            +.41291646+02


18.  ******** DUMP REQUEST AT 30., REL LOC 00022, ABS LOC 03300, IN DECK PF2021

              (DISC,DISC,F)

PF2021       DISC         SOLVE +88
00130 03406 DISC         +.00000000+00

              (A,A,F)

PF2020       A            COM1
00235 03242 A            +.40000000+01

              (B,B,F)

PF2020       B            COM1  +1
00236 03243 B            -.20000000+02

              (C,C,F)

PF2020       C            COM1  +2
00237 03244 C            +.25000000+02

              (DISC,DISC,F)
```

```
PF2021       DISC        SOLVE +88
00130 03406 DISC        +.10000000+00


19.  ******** DUMP REQUEST AT 30., REL LOC 00022, ABS LOC 03300, IN DECK PF2021

                (DISC,DISC,F)

PF2021       DISC        SOLVE +88
00130 03406 DISC        -.16000000+02

                (A,A,F)

PF2020       A           CON1
00235 03242 A           +.15000000+01

                (B,B,F)

PF2020       B           CON1  +1
00236 03243 B           -.12000000+01

                (C,C,F)

PF2020       C           CON1  +2
00237 03244 C           +.11000000+01

        579 LINES OUTPUT.
```

```
$IBDBL
*DEBUG PF0300  1
      NAME  PIP/=NEW(D)
      IF (PI .LE. PIP)  ON 5  GO TO 2
      SET PIP = PI
      SET C = 10.0
      RETURN
    2 DUMP PIP, PI, B, N, I
      DUMP C,D
      CALL EXIT
*DEND
```

```
         PF0300

      DOUBLE PRECISION  PI, B, N
      I=100                                                    ,1
      B=1.0                                                    ,2
      C=5.0                                                    ,3
      N=6.D0                                                   ,4
      WRITE (6,100)                                            ,5    ,6
      GO TO 2                                                  ,7
    1 I=I-1                                                    ,8
      N=N*2.D0                                                 ,9
      B=DSQRT((1.-DSQRT(1.-B*B/4.))**2+B*B/4.)                 ,10
    2 PI=N*B*.5                                                ,11
      D= 2.0*C                                                 ,12
      WRITE (6,101) N,B,PI                                     ,13   ,14   ,15
      IF (I.GE.0) GO TO 1                                      ,16   ,17   ,18
      RETURN                                                   ,19
  100 FORMAT (1H1,4X 15HNUMBER OF SIDES 11X 14HLENGTH OF SIDE 9X
     114HESTIMATE OF PI / 1X)
  101 FORMAT (1XD23.16,2XD23.16,2XD24.16)
      END                                                      ,20
```

```
LISTING OF DEBUGGING DUMPS

1,    ******** DUMP REQUEST AT 1., REL LOC 00031, ABS LOC 03011, IN DECK PF0300

                    (PIP,.IBDB4+1,D)

.DBGTX       PIP           .IBDB4
00225 17355  PIP          +.31415926535897790+01

                    (PI,PI+1,D)

PF0300       PI            ......+122
00172 03152  PI           +.31415926535897790+01

                    (B,B+1,D)

PF0300       B             ......+124
00174 03154  B            +.19505574390228700-08

                    (N,N+1,D)

PF0300       N             ......+126
00176 03156  N            +.32212254720000000+10

                    (I,I,X)

PF0300       I             ......+128
00200 03160  I                  +71.

                    (C,C,F)

PF0300       C             ......+129
00201 03161  C             +.10000000+02

                    (D,D,F)

PF0300       D             ......+130
00202 03162  D             +.20000000+02

          199 LINES OUTPUT.
```

NUMBER OF SIDES	LENGTH OF SIDE	ESTIMATE OF PI
0.6000000000000000D 01	1.0000000000000000D 00	0.3000000000000000D 01
0.1200000000000000D 02	0.5176380902050415D 00	0.3105828541230249D 01
0.2400000000000000D 02	0.2610523844401031D-00	0.3132628613281236D 01
0.4800000000000000D 02	0.1308062584602860D-00	0.3139350203046865D 01
0.9600000000000000D C2	0.6543816564355224D-01	0.3141031950890507D 01
0.1919999999999999D 03	0.3272346325297353D-01	0.3141452472285459D 01
0.3840000000000000D 03	0.1636227920787424D-01	0.3141557607911854D 01
0.7680000000000000D 03	0.8181208052469571D-02	0.3141583892148315D 01
0.1536000000000000D 04	0.4090612582328185D-02	0.3141590463228046D 01
0.3072000000000000D 04	0.2045307360676606D-02	0.3141592105999266D 01
0.6143999999999999D 04	0.1022653814027393D-02	0.3141592516692152D 01
0.1228800000000000D 05	0.5113269237248337D-03	0.3141592619365378D 01
0.2457600000000000D 05	0.2556634639513090D-03	0.3141592645033684D 01
0.4915200000000000D 05	0.1278317322367660D-03	0.3141592651450760D 01
0.9830400000000000D 05	0.6391586615102192D-04	0.3141592653055029D 01
0.1966079999999999D 06	0.3195793307959082D-04	0.3141592653456096D 01
0.3932160000000000D 06	0.1597896654030539D-04	0.3141592653556362D 01
0.7864320000000000D 06	0.7989483270216441D-05	0.3141592653581428D 01
0.1572864000000000D 07	0.3994741635116187D-05	0.3141592653587694D 01
0.3145728000000000D 07	0.1997370817559089D-05	0.3141592653589260D 01
0.6291456000000000D 07	0.9986854087796690D-06	0.3141592653589651D 01
0.1258291200000000D 08	0.4993427043898500D-06	0.3141592653589748D 01
0.2516582400000000D 08	0.2496713521949269D-06	0.3141592653589772D 01
0.5033164800000000D 08	0.1248356760974637D-06	0.3141592653589778D 01
0.1006632960000000D 09	0.6241783804873187D-07	0.3141592653589779D 01
0.2013265920000000D 09	0.3120891902436593D-07	0.3141592653589779D 01
0.4026531840000000D 09	0.1560445951218296D-07	0.3141592653589779D 01
0.8053063680000000D 09	0.7802229756091482D-08	0.3141592653589779D 01
0.1610612736000000D 10	0.3901114878045741D-08	0.3141592653589779D 01
0.3221225472000000D 10	0.1950557439022870D-08	0.3141592653589779D 01

(BUFFER AT 20501)

DUMP NUMBER 00001

6.3.9 System Debugging Aids
Jul/65

6.3.9.1 Error Walk-Back Feature
Jul/65

The Subroutine Library provided with the FORTRAN IV compiler contains a debugging aid called the error walk-back feature. All the subroutines of this library call the system subroutine FXEM when they detect an error condition caused by the user's object program. FXEM then gives a trace which lists the sequence of calls in reverse order through all levels of subprograms used back to the main program. FXEM is called when:

1) A mathematical subroutine is called with an illegal argument such as SQRT (-25.97), ALOG (-15.3), EXP (156.1), etc.
2) An input/output error occurs.
3) An invalid value for a computed GO TO has been found.

Three pieces of information are given for every CALL statement in the sequence:

1) The name of the routine in which the CALL statement occurs.
2) The absolute location of the CALL in core storage.
3) The line or identification number of the CALL statement as it appears in the compilation listing of the given routine.

An error flow trace resulting from an input/output error or a computed GO TO error causes the job to be terminated. Execution of a job is resumed after an error flow trace resulting from incorrect usage of a mathematical subroutine. However, a job will be terminated if 50 error flow traces exist. Most messages associated with an error flow trace are self-explanatory, but some which pertain to an input/output error need further explanation. They are:

1) PHYSICAL RECORD SIZE EXCEEDS BUFFER SIZE
   This usually means that the programmer is reading or writing a binary tape and by mistake is using logical tapes 7, 9, or 11 instead of 8, 10, or 12. (See section 8.4.0).
2) LIST EXCEEDS LOGICAL RECORD LENGTH
   This message is given when a READ statement for a binary tape has a list longer than the list used when the tape was written. If the tape has variable length records, then having the tape at an incorrect position could cause this message.
3) LOGICAL UNIT NOT DEFINED FOR VALUE xx
   This means that a logical tape number is being used which is not 3, 5, 6, 7, 8, 9, 10, 11, or 12.
4) WRITE REQUEST ON UNIT DEFINED AS SYSIN1 ILLEGAL
   Tape 5 is being used as an output tape instead of an input tape.
5) READ REQUEST ON UNIT DEFINED AS SYSOU1 ILLEGAL
   Tape 6 is being used as an input tape instead of an output tape.

6.3.9.1 Jul/65 (continued)

6) ILLEGAL CHAR FOR L CONVERSION IN DATA BELOW
An L conversion appears in a format statement, and the variables being used with it do not contain the logical constants TRUE or FALSE.

6.3.9.2 Jul/65 Floating Point Overflow

If during execution of a job a computed quantity becomes too large (i.e., exceeds 10^{38}), the following message is printed:

FLOATING POINT OVERFLOW AT locn

where locn is the absolute location of the overflow. The programmer can determine in which deck the overflow occurred by using the Memory Map (explained in section 6.5.0). The equation causing the overflow can then be located by using the Reference to Defined Symbols Table (also explained in section 6.5.0) which accompanies the compilation of the above deck.

6.3.9.3 → Nov/65 Input/Output Error Messages Generated by the 7040

All the input and output for a job is performed by the 7040. The following messages are given by the 7040 when information on the disk or a tape is being read incorrectly:

1) UNITxx READING BEYOND VALID INPUT
   a) This means the job is trying to read non-existent information from the disk. Usually this is caused by an attempt to read more records or more files than have been written on the disk. If the information was written on the disk by means of the $SETUP card, then an incorrect file count on the $SETUP card could have caused this message.

   b) In order to determine which logical unit was being used at the time of difficulty, the number of units utilized must be known. This includes units for which $SETUP cards have been submitted as well as scratch units (i.e., those without $SETUP cards). The logical units with $SETUP cards are assigned a name by 7040 first and the scratch units next.

   Example: Assume a job is using 6 logical units, 3 with $SETUP cards and 3 scratch units, then:

NAME ASSIGNED	REFERS TO
UNIT xx	
UNIT05,	the 1st logical unit having a $SETUP card
UNIT06,	the 2nd " " " " " "
UNIT07,	the 3rd " " " " " "
UNIT08,	the 1st logical scratch unit used
UNIT09,	the 2nd " " " "
UNIT10,	the 3rd " " " "

2) CODE xx PERMANENT READ REDUNDANCY
This means the job is reading a tape incorrectly. Usual causes of this failure are:
   a) The tape is not in the correct position.
   b) The tape is being read in the wrong mode.
   c) The wrong tape has been mounted.

6.4.0 Jul/65

Job Termination

Any job operating under the IBJOB monitor may be terminated by any of the following techniques:

1) CALL EXIT

2) CALL DUMP

3) CALL POST (If AL CRASH has been used)
The detailed writeups DUMP and POST are available in the Ames Fortran IV Subroutine Manual.

4) STOP
STOP transfers control to the control subroutine .LXCON.

5) There is no more data on the input tape when a READ (5,n) statement is encountered (n is a format number). When this occurs, the routine ENDFIL is called. If the programmer has written a subroutine with this precise name (ENDFIL), then control is transferred to this routine which must at its conclusion CALL EXIT. If there is no such routine, the existing ENDFIL calls EXIT and the job is terminated.

6.5.0
Jul/65

Description of IBJØB Output

The user is supplied with a two-ply listing of his output. Included therewith are compilation listings, tables of symbols, a memory map, an accounting record, and information supplied by the monitor system. The various parts of a typical output listing are shown below.

Item

$JOB Card

Monitor System Messages

Debugging Packet Listing

Compile Listing } Repeated for each compile

Symbol Reference Data }

Memory Map

Output, Computed Results (Logical tape 6)

Debugging Output

Accounting Record

Output, Computed Results (Other than tape 6)

$JOB Card

The $JOB card, which identifies the job, is printed as the first line of the output. The user's name thus conveniently appears on the first page of his printout.

Monitor System Messages

The system supplies the following items, immediately after the $JOB card:

1) The number of lines of output for this job.

2) The position of peripheral files (tapes) at the end of the job. This position is given by number of records and files read or written on the extra tapes. This may be of significant assistance in debugging.

3) A one-line message for any error trace occurring. The error walk-back itself appears in the appropriate position within the normal computer output.

Listings of Compilations

The compilation listing consists of source statements and associated internal formula numbers. If diagnostic messages occur, they are inserted following the appropriate source language statement.

(continued)

Symbol Reference Data

Two tables of symbols are printed following each compile, and their functions are described below.

References to Defined Symbols

This table contains all the symbols defined in a given compilation. It includes those named in the FORTRAN source program, as well as those generated by the compiler. All format numbers, statement numbers, variables, and common block names can be found in the column labelled SYMBOL as follows:

1) Format numbers and statement numbers are listed first and appear with a trailing ".". (For example, 1, 200, and 4000 appear as 1., 200., and 4000.).

2) All variables including those in the source program are listed next and are in alphabetical order.

3) Common block names appear in the variable list and are identified by the symbol LCTR in the CLASS column.

4) All other symbols appearing in the SYMBOL column are those generated by the compiler. Thus, this list not only provides the programmer with a useful check for statement numbers and variable names but is also useful for finding possible keypunching errors.

An octal number appears in the column headed VALUE and is associated with all symbols except those naming Labelled Common. It has the following interpretation:

1) Statement number - relative storage location of the first machine language instruction for that statement.

2) Variables - relative storage location of that variable.

Absolute locations for the above can be determined by adding the relative octal location to the origin of the subroutine as given in the Memory Map. Such information is necessary to determine the location of a floating point overflow or to determine the storage locations in an octal dump.

Since a Labelled Common block is shared by more than one subroutine, the loader assigns storage for this block to the first subroutine loaded which has reference to it. The absolute location of the block can be determined by using the reference table of the appropriate subroutine and the Memory Map.

Blank Common is loaded last, and the location of the first Common variable is given in the Memory Map by //. To obtain the absolute location of any other Blank Common variable, the programmer uses the location of // and the location of the desired variable relative to the beginning of Common. This last quantity is the difference of the locations of the desired variable and the first blank common variable, as listed in the reference table of any program containing them.

References to Virtual Symbols

The SYMBOL column of this table provides the programmer with a list of all the subroutines this program uses including those required from the library tape. In addition it lists a control section for each logical tape required in the program, e.g., .UN05., .UN06., .UN08., etc. The control section names which are enclosed within decimal points refer to subroutines required by the compiler.

There are other numbers included in these two tables which have not been explained since they are related to the final assembly (machine language) version of the program.

The Memory Map

The Memory Map is produced by the loader (IBLDR) as it loads the program into core and consists of (1) a list of the tape units (by logical number) required by the program, (2) a statement of the amount of storage used by the object program, exclusive of buffers, and (3) a list of the decks and subroutines that comprise the object program. The absolute location of the origin for each deck or subroutine is given, as well as the deck name. Unlabelled Common variables are in a control section named //, which appears last in this list. At the bottom of the Memory Map, the area of core reserved for input-output buffers is specified, as is the area of core not used by the program.

If the loader encounters any difficulties in loading the object program, e.g., if a required subroutine is missing, a message to this effect is printed after the Memory Map. If the program is too large to fit into core storage or if after loading there is insufficient storage for input-output buffers, appropriate messages are printed. In these situations the programmer must take steps to shorten the program or to use the overlay procedure.

Output of Computed Results (Logical Tape 6)

This printout will be as specified by output statements in the object program. It may also include error walk-backs. (A job is automatically terminated after 50 error walks.) If dynamic dumping has been requested, the message DUMP NUMBER________ is printed each time a dump occurs. This number also appears at the beginning of the actual dump information with the debugging output.

Debugging Output

Examples of debugging output are given in section 6.3.8.2 of this manual. This output follows the standard tape 6 output and precedes any other printed information.

Accounting Record

The accounting record consists of a printout of the programmer's name, CAB number of the program, the J. O. number, the date, the compile and load time, and the execution time. The two times are accurate to 0.01 minute.

Preceding the accounting record and following all other printed information, the number of lines of output is printed.

Output of Computed Results (Extra Tapes)

Any printed output requested on tapes other than logical tape 6 will appear following the accounting record. It is to be noted that the total number of lines of output, as printed ahead of the accounting record, includes the printout from these extra tapes.

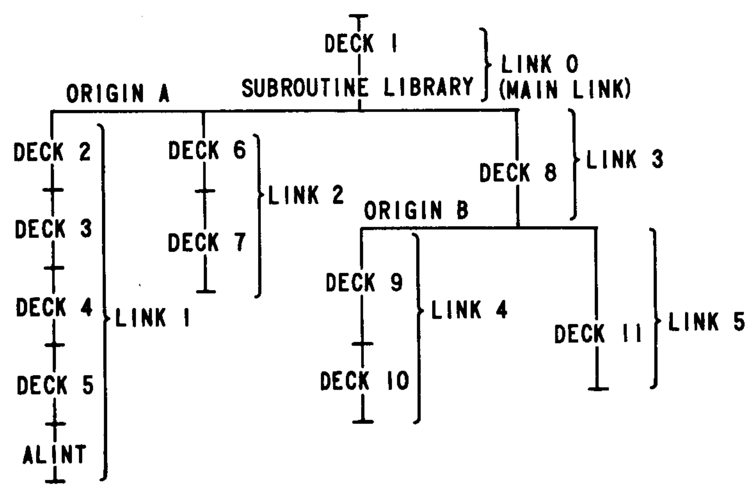
6.6.0 Nov/65 Overlay (Storage Allocation for Large Jobs)

6.6.1 Nov/65 Purpose

The overlay feature of the loader (IBLDR) of the IBJOB monitor provides a means by which different parts of a program may utilize the same areas of storage. In this way the jobs which exceed the limits of available storage may be executed. The program is first divided into groups of subprograms (called links). These links are then arranged by the programmer so that non-interacting links use the same storage area. Once this has been done, the proper use of control cards causes the loader to form what is called the overlay tape (at Ames, the disk is used).

Full details for the use of the overlay feature may be found on pages 29-33 of the IBM publication IBM 7090/7094 IBSYS Operating System, IBJOB Processor which have been reproduced for distribution to programmers.

6.6.2 Nov/65 Description



THE OVERLAY STRUCTURE

The sketch above illustrates in a schematic way the structure of an overlay job. The six links of this job are shown as vertical lines. Each link is composed of one or more individual subprograms (decks). Link 0, which includes the required programs from the Subroutine Library Tape, is in core at all times. The loading of the other links is effected by calling a subroutine belonging to that link. A rule in the calling of links is that the link being called may not overlay the calling link. Thus it can be seen from the sketch that a given link may call one that is above it or below it in the "tree", but may not call one that starts from the same origin. That is:

(a) A subroutine in link 0 can call a subroutine in any link, since it is always in core, and is never destroyed by the other links.

6.6.2 (continued)
Nov/65

(b) A subroutine in link 3 may call a subroutine in link 4 or link 5.

(c) A subroutine in link 2 may not call a subroutine in any other link except link 0.

(d) A subroutine in link 4 may not call a subroutine in link 5, but a subroutine in either link 4 or 5 may call a subroutine in link 3 or link 0.

It is to be noted that if a subroutine in link 3 calls a subroutine in link 4, and then subsequently calls a subroutine in link 5, then link 5 overlays link 4. When a given link is in core, then all of the links <u>above</u> it are also in core. Thus, when link 5 is in core, link 3 and link 0 are in core. Note in the sketch that the Library Tape Subroutine AL INT is shown in link 1 and not link 0. This is accomplished by use of a $INCLUDE control card in link 1, which has AL INT in its variable field.

6.6.3 <u>Deck Makeup</u>
Nov/65

```
    $JOB
    $IBJOB
*→
    $DECK     DECK1     }   LINK 0
    $ORIGIN   A
*→
    $DECK     DECK2
    $DECK     DECK3         LINK 1
    $DECK     DECK4
    $DECK     DECK5
    $INCLUDE  ALINT
    $ORIGIN   A
*→
    $DECK     DECK6
    $DECK     DECK7         LINK 2
    $ORIGIN   A
*→                          LINK 3
    $DECK     DECK8
    $ORIGIN   B
*→                          LINK 4
    $DECK     DECK9
    $DECK     DECK10
```

6.6.3 Nov/65 <u>Deck Makeup</u> - (continued)

```
    $ORIGIN   B      }
*→                   }    LINK  5
    $DECK     DECK11 }
    $DATA
```

*→ Special OVERLAY tab cards upon which should be specified the deck makeup for each link.

The deck makeup of the previous example is shown above. The $DECK cards within each link cause the indicated binary decks to be obtained from disk. Source decks to be compiled or assembled may also be included within a link and they are inserted ahead of the $DECK cards. The decks and or $DECK cards for each link exeept link 0 are preceded by a $ORIGIN card which names the origin for that link. An OVERLAY bab card obtainable in room 129 of the Data Reduction Building follows the $IBJOB card and each $ORIGIN card. The programmer uses this card to list all the decks belonging to each link.

→ 6.6.4 Nov/65 <u>Note</u>

The first link determines which level of IOCS will be loaded. If "REWIND N" and/or "ENDFILE N" statements are used in the subsequent links, but not in the first, minimum IOCS will be loaded without the necessary routines; therefore "BASIC" must be specified in the variable field of the $IBJOB card (columns 16-20) in order that the job execute correctly.

6.7.0 MAP (Macro Assembly Program)
Jul/65

6.8.0
Jul/65

COBOL (COmmon Business Oriented Language)

7.0.0 Jul/65 — FORTRAN MONITOR SYSTEM (FMS)

7.1.0 Jul/65 — Function

The FORTRAN MONITOR controls the FORTRAN II compiler and FAP (the Fortran Assembly Program). In addition, if a job is to execute, it loads all the object decks and then searches the library tape for any missing subroutines which are required. These are then loaded and execution of the job is begun.

7.2.0 Jul/65 — FORTRAN II

7.2.1 Nov/65 — Job Makeup

A FORTRAN II job which writes a plot tape and contains debug macro cards has the following composition:

1) PLOT card (solid blue)

2) EOF card (color/catagory)

3) $JOB card (solid orange)

4) $SETUP (solid orange)

5) $EXECUTE FORTRAN (solid orange)

6) *XEQ card, if Load and Go (solid orange)

7) *LABEL, if FORTRAN compile (solid orange)

8) *SYMBOL TABLE (solid orange)

9) FORTRAN II source deck (brown stripe)

10) END card (grey stripe)

(Items 7, 8, 9, and 10 may be repeated as often as necessary for compiling all decks.)

11) *DEBUG card (solid red)

12) DEBUG MACRO cards (solid red)

→ Note: Two * cards (solid orange) must precede the $DECK cards if a job does not contain debug macro cards.

13) $DECK cards (solid orange)

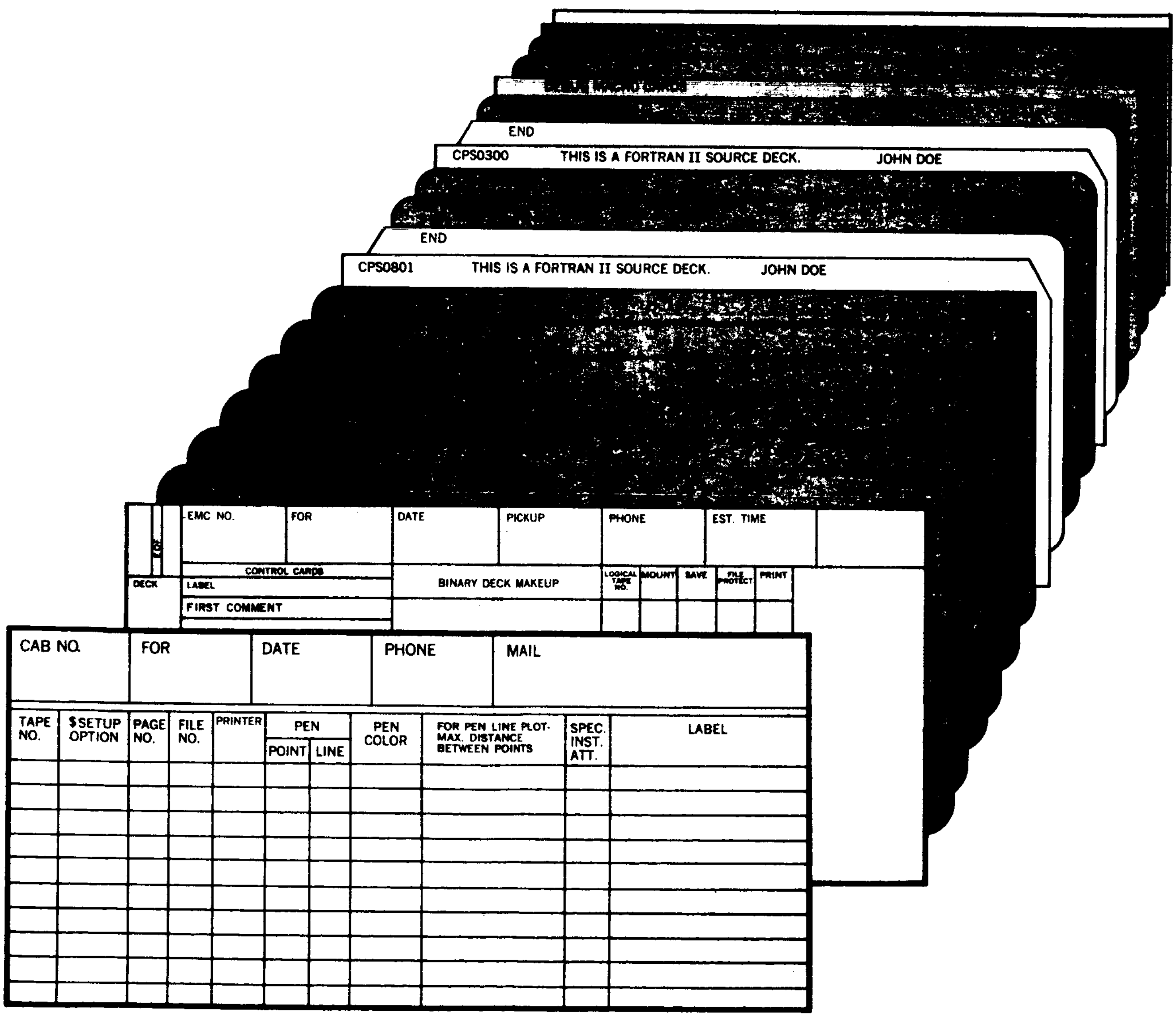
14) *DATA card (solid orange)

15) Input data cards, if required. (Color is optional, but solid orange cards and FORTRAN cards are excluded).

7.2.2
Jul/65

Sample Job Makeup

Sample Fortran II Job



7.2.3 Control Cards
Jul/65

7.2.3.1 PLOT card (solid blue)
Jul/65

This card is used first by the DCS operator and later by the EAI Dataplotter operator. It is the first card of every job which writes a plot tape. The following is an example showing its usage:

[illegible] NO. CAB	FOR	DATE	PHONE	MAIL
EP18YO	John Doe	6/30/65	2895	

TAPE NO.	$SETUP OPTION	PAGE NO.	FILE NO.	PRINTER	PEN POINT	PEN LINE	PEN COLOR	FOR PEN LINE PLAT- MAX. DISTANCE BETWEEN POINTS	SPEC. INST. ATT.	LABEL
7	Plot 1	1	1	✓		✓	Red	2.0		IDVSVI Study, Subjs. 1-12, 16, Table 1
			2			✓	Red	0.25		
			3			✓	Green	0.25		
		2	4	✓		✓	Red	2.0		IDVSVI Study, Subjs. 1-12, 16, Table 2
			5		✓		Red			
9	Plot 2	3	6	✓		✓	Red	2.0		IDVS2 Study, Subjs. 1-6, Tables 1 & 2
			7			✓	Red	0.25		
			8			✓	Green	0.25		
			9		✓		Red			

For a detailed explanation of this card see the section on Plotter Usage (8.5.0).

7.2.3.2 EOF Card
Jul/65

This card is used by the DCS operator and follows the blue PLOT card if one exists. It is prepunched with 7 and 8 in column one. The color of this card identifies the job category in the following way:

Neutral - Standard Monitor DEBUG or RECOM Job.

Solid Yellow - Standard Monitor PRODb Job.

Solid Brown - Special Monitor DEBUG or RECOM Job .

Solid Green - Special Monitor PRODb Job.

Solid Salmon - Non-Monitor Job.

Solid Red - Priority Run (Available only from S. M. Crandall of CAB)

7.2.3.2
Nov/65

(continued)

These cards are available in the operations office and <u>will</u> be prepunched. A new card must be provided with each job and will not be returned. It should be properly filled out with:

1) CAB number for the job. (This must be identical to the CAB number on the $JOB card).

2) Name

3) Date

4) Pickup

5) Phone

6) Estimated Time (Execute Phase Only)

→ 7) **Additional tape assignments other than 5 for input, 6 for output, scratch tapes, and print tapes. (See Magnetic Tape Usage, section 8.4.0)**

Other information on this card is given to provide the user with a checklist for deck makeup. If this card is omitted, the job will not be processed but will be returned to the user.

EOF	EMC NO.	FOR	DATE	PICKUP	PHONE	EST. TIME
	PS2127	John Doe	7/1/65		2895	3 min.

	CONTROL CARDS	BINARY DECK MAKEUP
DECK	LABEL	
	FIRST COMMENT	
	END	
→ JOB	ID	
	XEQ	
	DATA	
OTHER	LIST	
	LIST8	
	FAP	
	CHAIN	
	DATE	
	SYMBOL TABLE	

LOGICAL TAPE NO	MOUNT	SAVE	FILE PROTECT	PRINT
7	Plot			
8	1595	✓	✓	
10	Assign	✓		

IBM L54062

7.2.3.3 Jul/65 <u>$JOB Card, solid orange</u>

This card is used by the IBSYS monitor and follows the EOF card. If for any reason it is missing, the job will not be entered into the input queue. It has the same format as the IBJOB $JOB card described in section 6.2.3.3.

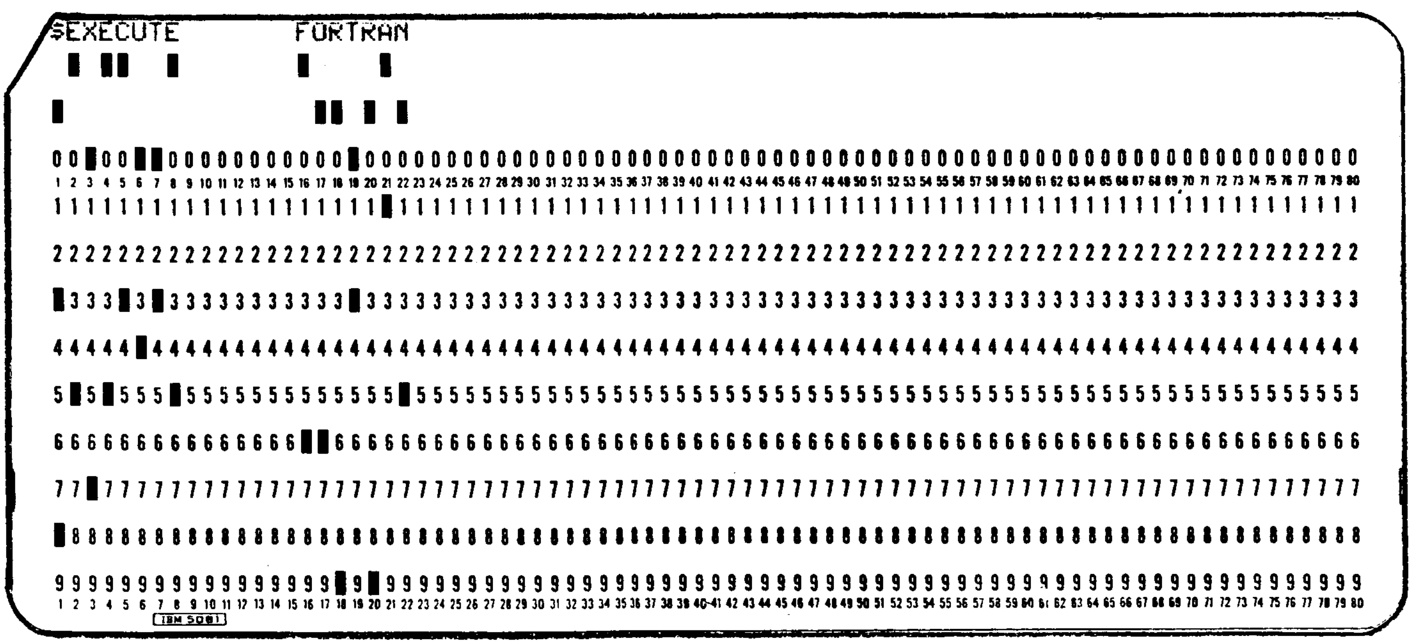
7.2.3.4 Jul/65 <u>$SETUP Card, solid orange</u>

All $SETUP cards follow the $JOB card, and they have the same format as the $SETUP cards described in section 6.2.3.4.

7.2.3.5 Jul/65 <u>$EXECUTE FORTRAN Card, solid orange</u>

Since each job must be a separate entity as it enters the card reader, it is necessary to name the exact system required for processing this job. If the job is a FORTRAN II job, as it is in this case, then a $EXECUTE FORTRAN card must call the FORTRAN Monitor System into action. This card must follow the $SETUP cards. It can be prepunched on a solid orange card and has the following format:

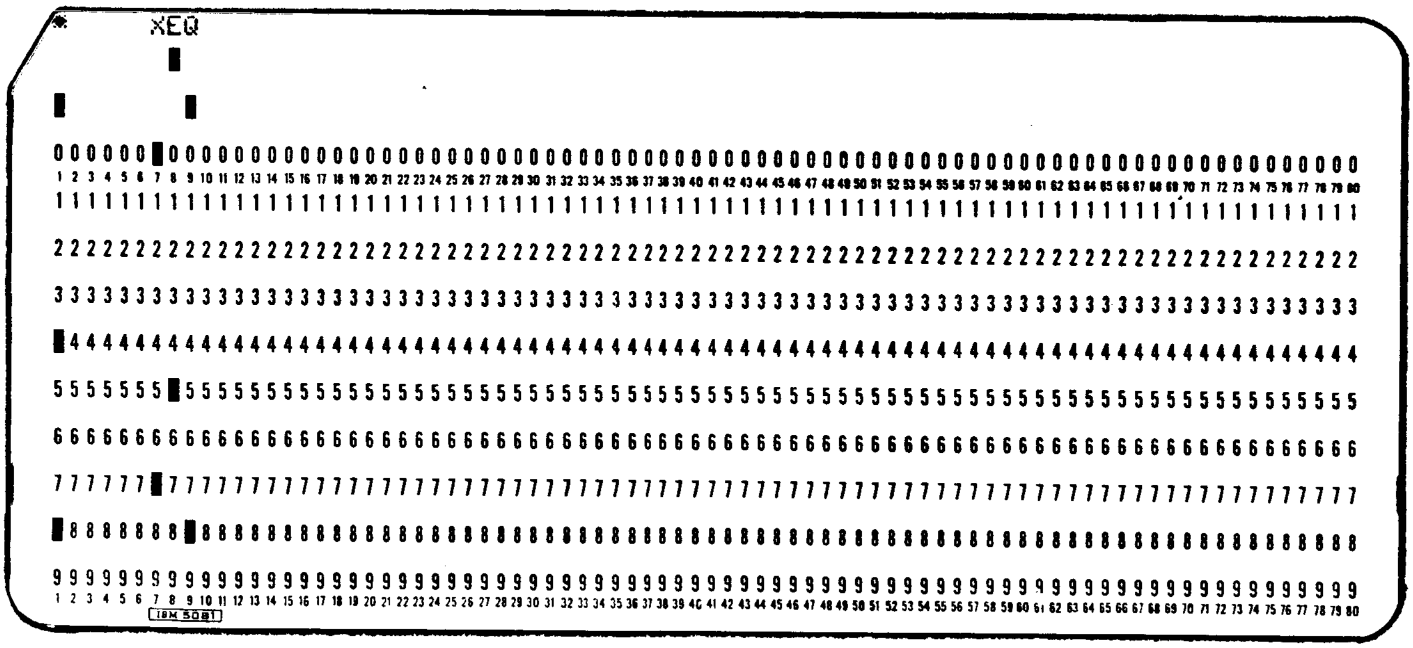
Column	Contents
1 - 8	$EXECUTE
16 - 22	FORTRAN



7.2.3.6 Jul/65 **XEQ Card, solid orange**

This card follows the $EXECUTE FORTRAN card and must be present in every job which is to have an execute, i.e., a compute phase. It is independent of the job classification, i.e., debug or production. If the job does not have an XEQ card, it will not go into an execute phase regardless of any other control card. The XEQ card which can be prepunched is always punched on a solid orange card and has the following format:

Column	Contents
1	Asterisk
7-72	XEQ (blanks ignored)



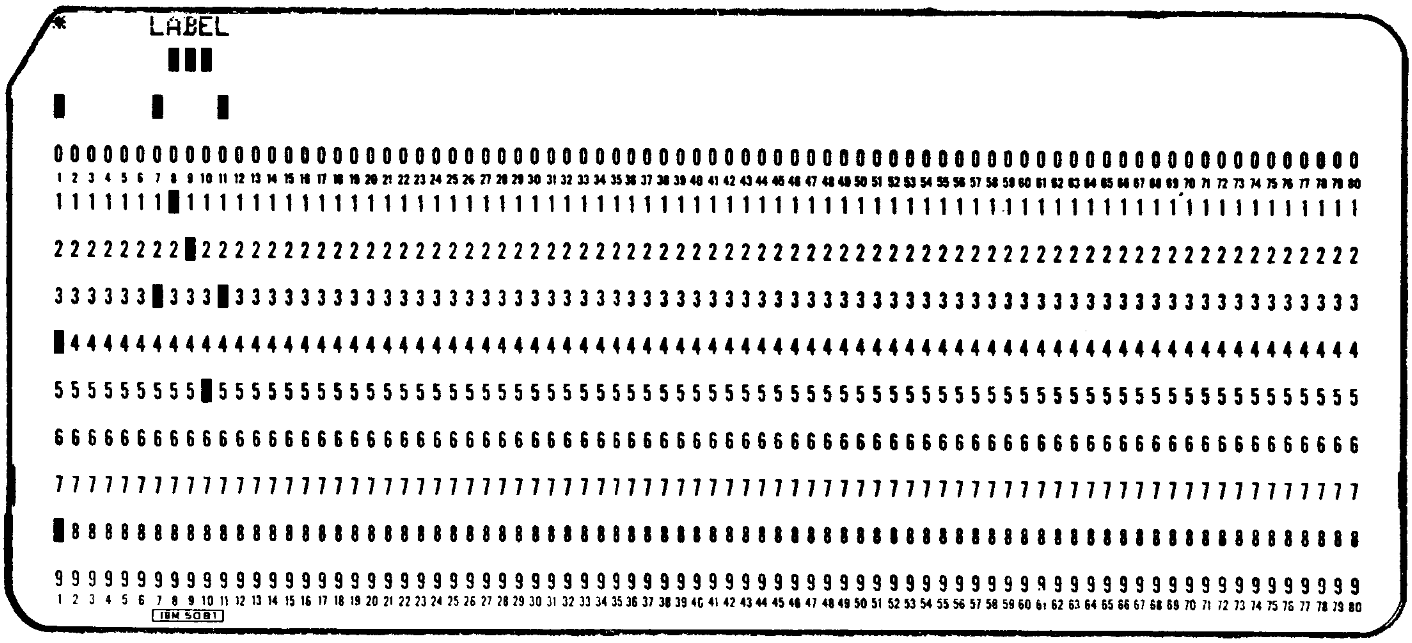
7.2.3.7 Jul/65 **LABEL, solid orange**

This card must occur as the first or second card of each deck within a job. It may be preceded only by a SYMBOL TABLE card. It gives the monitor the necessary information for labeling the binary decks in columns 73-78. The label which is used on the binary deck is the contents of columns 2-7 of the first card of each FORTRAN deck with a C in column one. Columns 79-80 contain the sequential numbering for the deck. The presence of this card is essential to assist the operator in separating binary decks. If a deck is not labeled, it will auto-

7.2.3.7 (continued)
Jul/65

matically be destroyed. It is not practical to operate under monitor control without this convenience. If the first comments card does not have anything in columns 2-7, the deck is incorrectly labeled and will again be destroyed. The LABEL card which can be prepunched will be keypunched on a solid orange card with the following format:

Column	Contents
1	Asterisk
7-72	LABEL (blanks ignored)

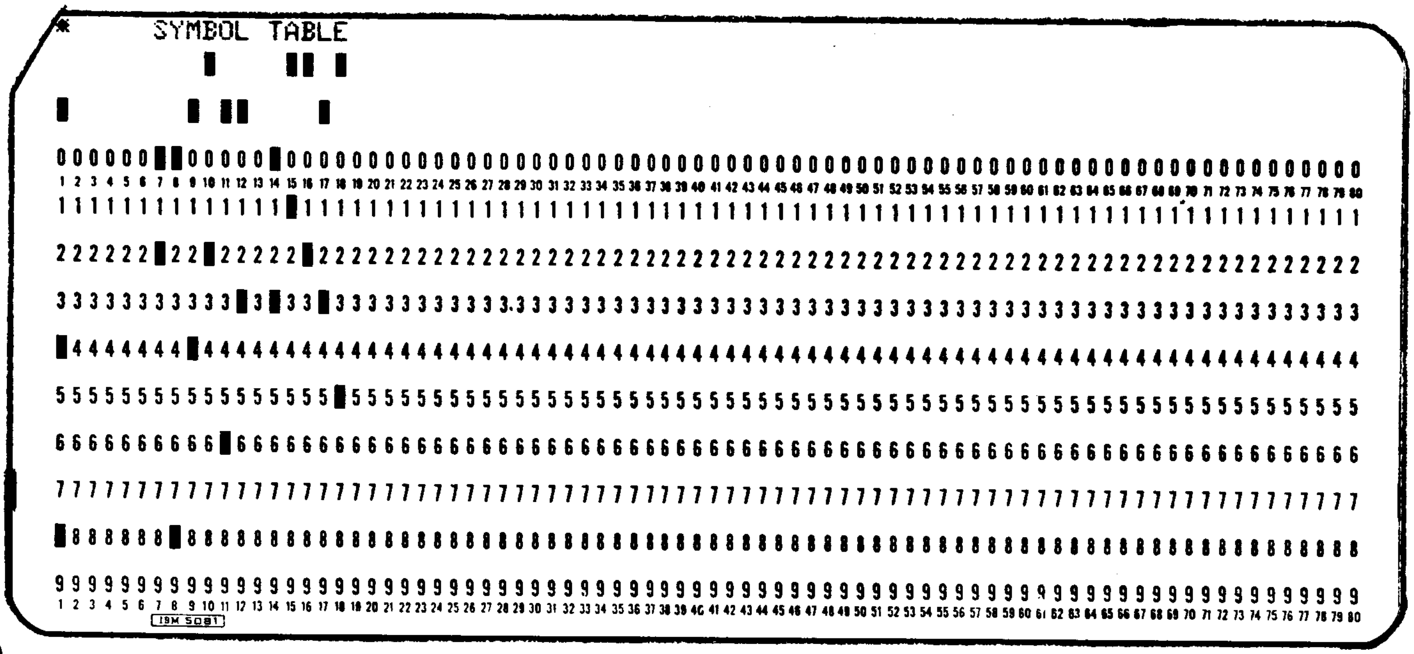


7.2.3.8 SYMBOL TABLE, solid orange
Jul/65

This card precedes the FORTRAN source desk to which it applies. It may precede or follow the LABEL card. It must be included in the compilation of any deck in which debug macros are to be inserted. It signals the compiler to add to the binary deck for that compilation all necessary information relating symbols and statement numbers to locations. This card must be punched on a solid orange card with the following format:

7.2.3.8 (continued)
Jul/65

Column	Contents
1	Asterisk
7-72	SYMBOL TABLE (blanks ignored)

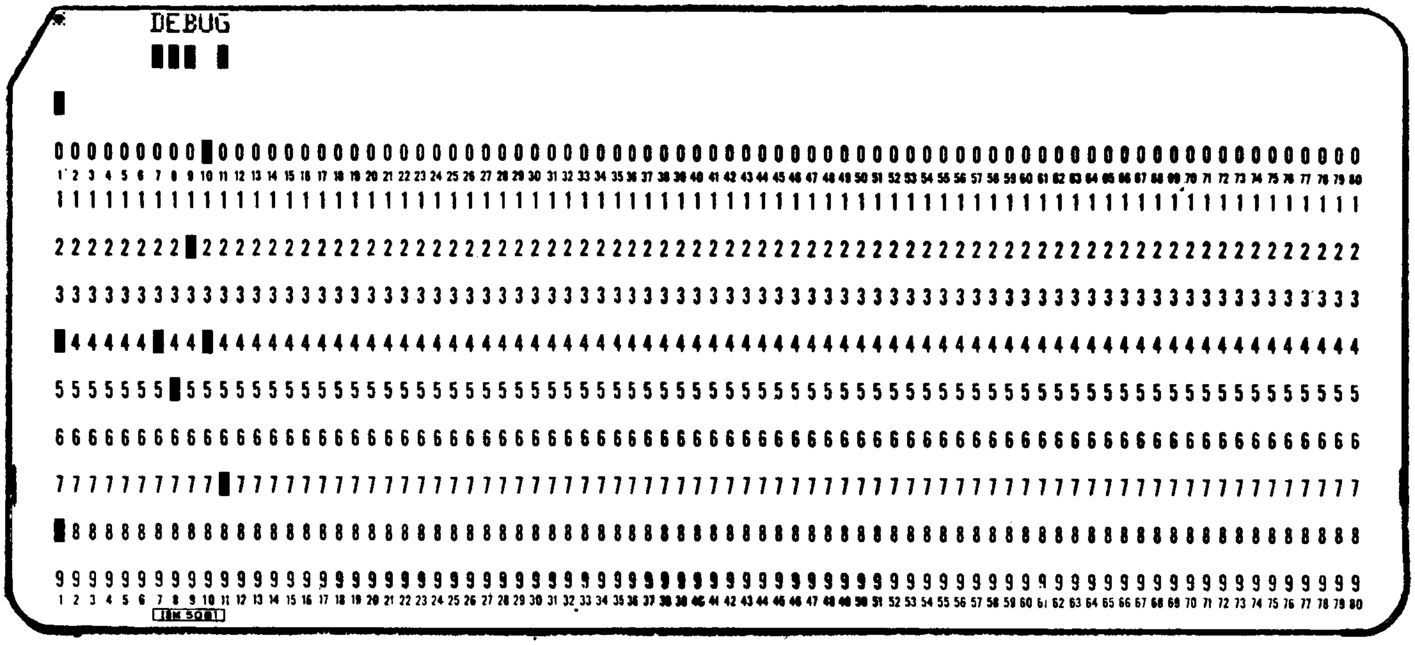


7.2.3.9 DEBUG, solid red
Jul/65

This card follows the last FORTRAN source program, if any is present, and precedes all debug macro cards for a given job. It indicates to the compiler that, at object time, debugging is requested and enables it to process the following debug cards. This card, which is necessary for every monitor job which requests debugging, may be prepunched. It is to be keypunched on a solid red card with the following format:

Column	Contents
1	Asterisk
7-72	DEBUG (blanks ignored)

7.2.3.9 (continued)
Jul/65

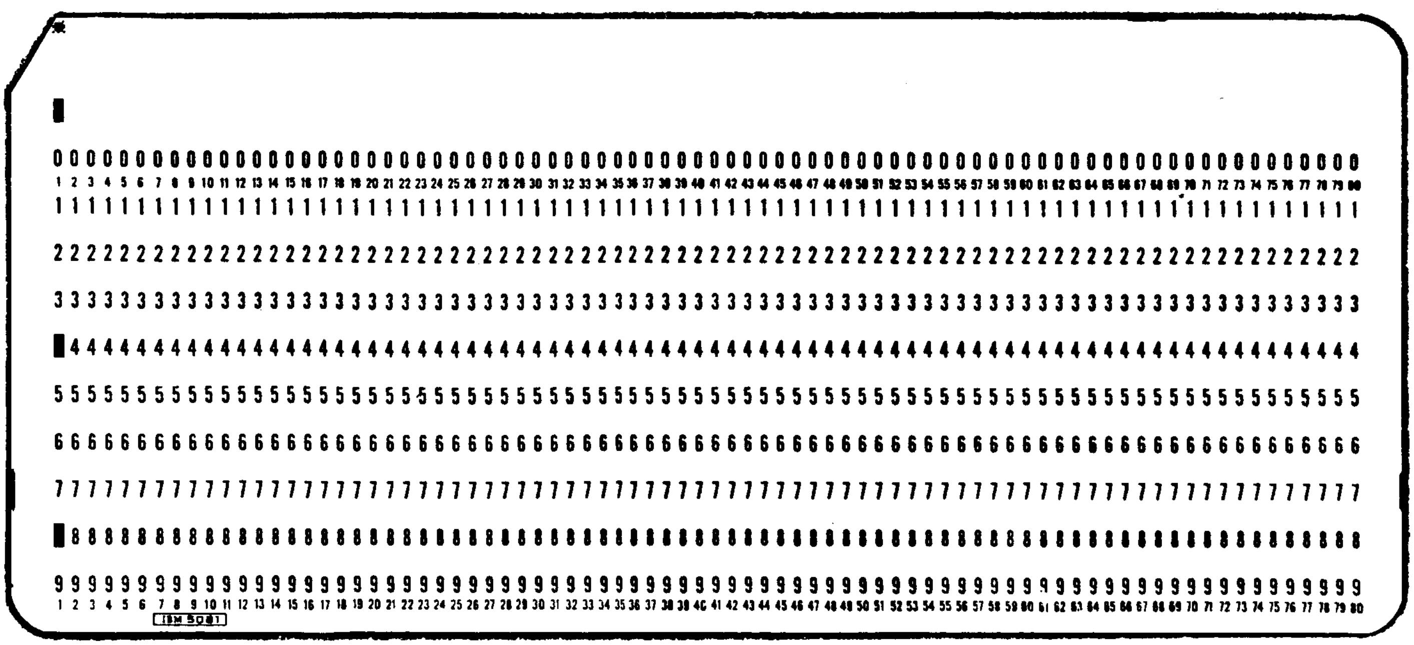


7.2.3.10 Additional * Cards
Nov/65

* Card, solid orange

* Card, solid orange

→ If a job does not contain debug macro cards it is necessary that two cards with an * punched in column 1 precede the $DECK cards.

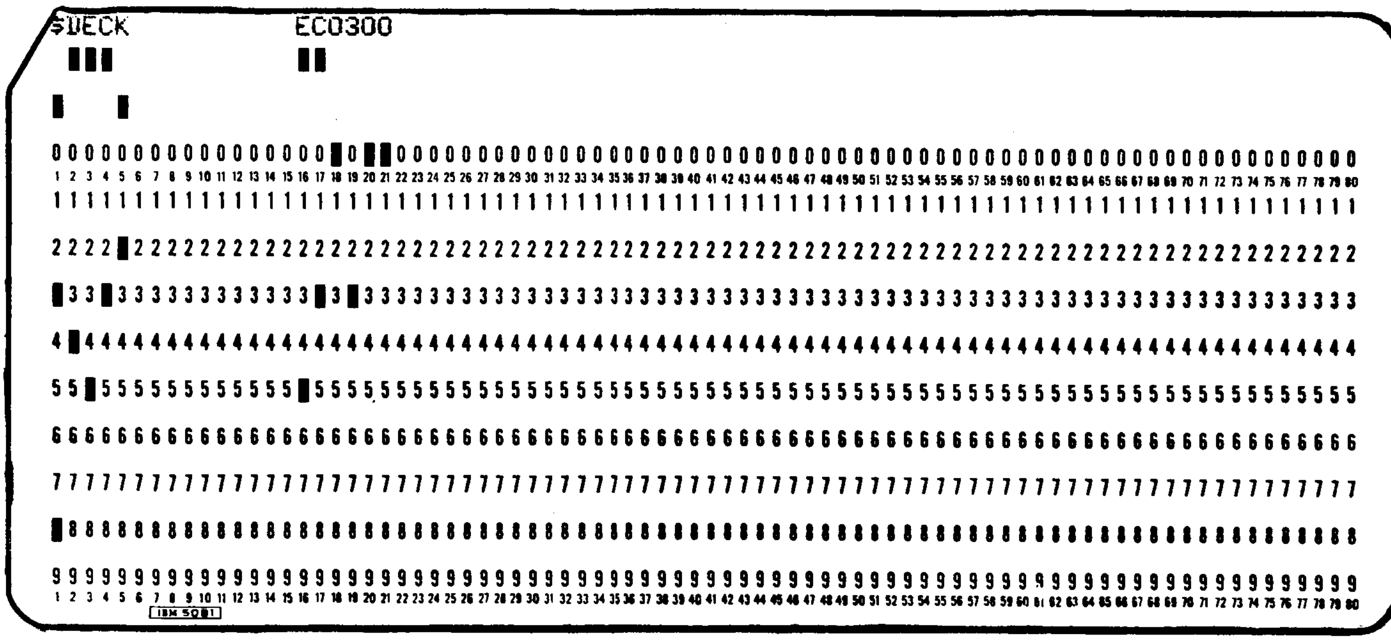


7.2.3.11 <u>$DECK, solid orange</u>

Nov/65

This control card is necessary when a binary deck is to be loaded from disk. There must be a $DECK card for each deck to be loaded from disk. The $DECK cards precede the DATA card and have the following format:

Column	Contents
1-5	$DECK
16-21	binary deck label



<u>Note on above</u>

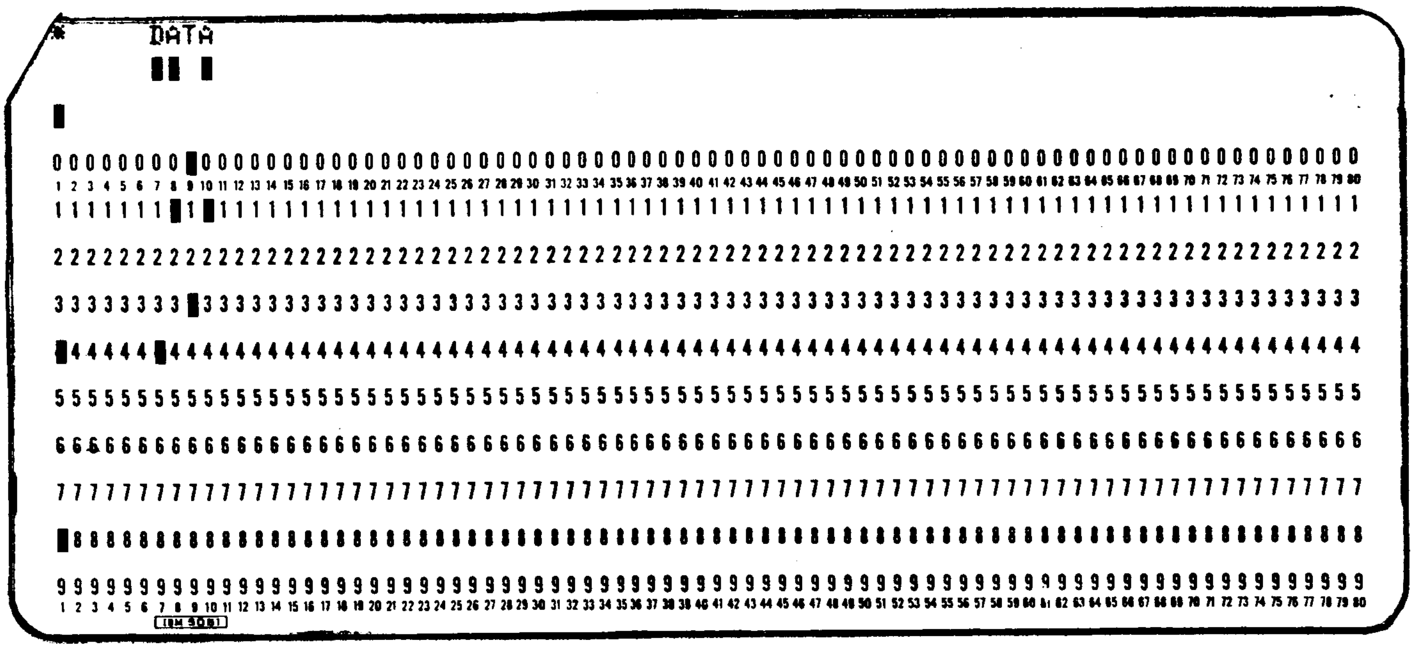
binary deck label - Binary decks belonging to the FORTRAN II Card Subroutine Library are labeled by ALSQ2, ALDER2, ALGAX2, etc. All other binary decks are labeled by the appropriate CAB number (EC0300, EC0301, PS0500, etc.)

7.2.3.12 Nov/65 DATA, solid orange

This card immediately precedes the data for the execute phase of the job and therefore is the last control card in the job. It signals the monitor when it has finished loading binary information and can start to execute.

The DATA card is necessary on every execute job and can be prepunched. It must be keypunched on a solid orange card with the following format:

Column	Contents
1	Asterisk
7-72	DATA (blanks ignored)



7.3.0 Jul/65 **Source Language Debugging at Object Time**

7.3.1 Jul/65 **Introduction**

In the current version of FORTRAN II there is included the facility of obtaining selective dumps of a FORTRAN program as it is being executed. The language for specifying these dumps is in FORTRAN type language, and dynamic dumping is achieved by interrupting the object program at the proper time, collecting the desired information, printing it, and then returning control to the object program. This option may be used in any main program or subprogram which has obtained the necessary information at compile time and may also be used with double-precision or complex arithmetic programs.

7.3.2 Jul/65 **Compilation**

To provide the object program with the information it must retain from the compilation phase, the control card SYMBOL TABLE must be added to the source deck. The format of this card is described in detail in 7.2.3.8. It precedes the FORTRAN program to which it applies.

7.3.3 Jul/65 **Execution**

To notify the monitor that selective dumping is to be effective on this job, a second control card, DEBUG, is added immediately following the last FORTRAN deck, if one is present, and ahead of the cards containing the dump information. All the cards which transmit information for dynamic dumping must precede the DATA control card. The format of the DEBUG card is described in 7.2.3.9.

7.3.4 Jul/65 **Debug Macro Cards**

7.3.4.1 Jul/65 **Name Card**

This card gives the name of the program to which the following debug macro

7.3.4.1 (continued)
Jul/65

cards apply and has the following format:

Column	Contents
1	N
7 - 72	Name of subprogram Blank if main program

7.3.4.2 Dump Cards
Jul/65

These cards stipulate the statement at which the dump is desired, the frequency of the dumps, the criterion for determining whether to dump, and the quantities to be dumped. They are to be punched on solid red cards, and the card format is

(continued)

Columns	Contents
1 - 5	Source statement number at which dump is desired
6	Continuation number as in FORTRAN
7 - 72	Field 1 $ Field 2 $ Field 3

Field 1 specifies frequency of dumps.

Its format is DUMP n_1 , n_2 , n_3 where n_1 , n_2 , n_3 are decimal integers. A dump will then occur the n_1th time the source statement is executed and every n_3rd time thereafter until the source statement has been executed n_2 or more times. If n_3 is omitted, it is assumed to be one as in DØ statements.

Field 2 specifies the criterion for dumping. Its format is:

IF $(V_1 \pm V_2)$ a_1, a_2, a_3

where

V_1, V_2 are single variables, either subscripted or non-subscripted, or decimal constants. Subscripts must be single unsigned integers. V_1 and V_2 may not have any additional signs used in the IF statement and must be of the same mode. The action of dumping takes place depending upon the words a_1, a_2, and a_3. If $(V_1 \pm V_2)$ is

(continued)

negative and a_1 = NO, no dump

a_1 = YES, dump

zero and a_2 = NO, no dump

a_2 = YES, dump

positive and a_3 = NO, no dump

a_3 = YES, dump

This field may be omitted entirely, i.e.

Field 1 $ Field 3

Field 3 specifies the locations to be dumped in the same notation used in FORTRAN. The following items are acceptable.

a. Subscripted or non-subscripted variables. Subscripts must be single unsigned integers.

   e.g. A(13), R, I, INDEX

b. Arrays of either mode, where the limits are specified in the following way:

   ARRAY (c - c')

   where ARRAY is the name of the array and c and c' are unsigned integers, c' > c. This will dump all elements from c to c' inclusive.

   e.g. A(3 - 10), I(6 - 14)

c. Common data, where this region can be defined either by COMMON DATA, which dumps the entire block, or COMMON DATA (S_1 - S_2)

7.3.4.2 Jul/65 (continued)

c. (continued)

where S_1 and S_2 are unsigned integers $S_2 > S_1$. This dumps the range of COMMON from S_1 to S_2 inclusive,

e.g. COMMON DATA (15 - 30)

d. Program data, where this region is defined by PROGRAM DATA which dumps all data of this subprogram that is not in COMMON.

7.3.5 Jul/65 Notes

7.3.5.1 Jul/65 Array Restrictions

In this debugging system arrays may only be one-dimensional. Therefore, if the array which is of interest is 2- or 3-dimensional, the elements must be located by the proper single unsigned integer. The proper value can be computed in the following way:

Given: array A of dimension (L, M, N).

To find: the (I, J, K) element as A(NO).

NO = I + (J - 1)L + (K - 1)(L)(M).

e.g. AR dimensioned (4, 3, 7).

to find AR(2, 2, 4) element.

NO = 2 + 1·4 + 3·4·3

= 2 + 4 + 36

= 42

AR(42) represents AR(2, 2, 4).

7.3.5.2
Jul/65

G-Format

All dumps are in a new format know as the G FORMAT. This format decides on I, E, or F format according to the following rules.

a. If a location contains information conforming to the integer format of FORTRAN, i.e., OXXXXXOOOOOO, the I format is used for printing.

b. Otherwise, the location is assumed to contain a floating point value. If the quantity is less than .1 or greater than $10^{\alpha+1}$(where the field is defined as $G\omega.\alpha$), E conversion is used; otherwise, F conversion is used.

c. Output is intermixed with programmed output and identified by subprogram name, statement number, and a count indicating the number of times the statement has been executed.

7.3.5.3
Jul/65

Timing

The following table describes the time at which a dump is taken for those FORTRAN statements which may be dumped.

Statement	Dump Occurs
Arithmetic	After evaluation
IF	After evaluation of expression but before transfer

7.3.5.3 Jul/65 (continued)

Statement	Dump Occurs
GO TO (unconditional and computed)	Before transfer
RETURN	Before transfer
CONTINUE	Before execution
INPUT/OUTPUT	Before execution

7.3.5.4 Jul/65 Placement of DUMP Cards

DUMP cards may not be placed on the following FORTRAN statements

CALL

DO

GO TO (assigned)

7.3.5.5 Jul/65 Storage Requirements at Execute Time

Number of extra locations = 60 + 2 * number of different statements at which dumps are requested

+ 26 for each DUMP card without an IF

+ 32 for each DUMP card with an IF

+ storage for each item in Field 3 as follows:

each single variable	6
each array	23
each COMMON data or Program data	15

7.3.5.6
Jul/65

Table Limits

Dumps may occur in no more than 20 subprograms. Maximum number of different statements at which dumps may occur in any one program or subprogram is 10.

Maximum number of different statements at which dumps may occur for a program and all its subprograms is 25.

Maximum number of Symbol Table entries per program or subprogram is 500.

7.3.5.7
Jul/65

Output Limit

There is a limit of 1000 lines of debug output.

7.3.5.8
Jul/65

CONTINUE Statements

CONTINUE cannot be used as a dummy statement on which to insert a debug macro. A = A is a better statement.

7.3.5.9
Jul/65

Use of Names in Debug Macros

To dump a variable by means of a debug macro the name of that variable must occur in the symbol table for that particular routine. This condition is satisfied only if

7.3.5.9 (continued)
Jul/65

the name of the variable appears in an executable statement in the routine. It is not sufficient that it appear in the specification statement of COMMON or DIMENSION.

7.3.6 Sample Program
Jul/65

```
      COMMON A,B,C,X1R,X1I,X2R,X2I
   11 NPAGE=1
    5 WRITE OUTPUT TAPE 6,8,NPAGE
    8 FORMAT(114H1            A                   B                   C                  X1
     1REAL          X1 IMAG          X2 REAL          X2 IMAG          PAGE I4//)
      NPAGE=NPAGE+1
      LINES=0
   14 READ INPUT TAPE 5,15,A,B,C
   15 FORMAT(3F10.5)
   20 IF (A) 16,17,16
   17 WRITE OUTPUT TAPE 6,18
   18 FORMAT(13H JOB FINISHED)
      GO TO 14
   16 CALL SOLVE
      IF (X1I) 90,91,90
   91 WRITE OUTPUT TAPE 6,95,A,B,C,X1R,X2R
   95 FORMAT(1H ,1P4E15.4,E30.4)
      GO TO 19
   90 IF (X1R) 100,101,100
  101 IF (X2R) 100,102,100
  102 WRITE OUTPUT TAPE 6,103,A,B,C,X1I,X2I
  103 FORMAT(1H ,1P3E15.4,2E30.4)
      GO TO 19
  100 WRITE OUTPUT TAPE 6,110,A,B,C,X1R,X1I,X2R,X2I
  110 FORMAT(1H ,1P7E15.4)
   19 LINES=LINES+1
      IF (LINES-40) 14,5,5
      END(1,0,0,0,0,1,1,0,0,0,0,0,0,0,0)
```

```
      SUBROUTINE SOLVE
      COMMON A,B,C,X1R,X1I,X2R,X2I
   30 DISC=B**2-4.0*A*C
      IF (DISC) 50,60,70
   50 X1R=-B/(2.0*A)
      X2R=X1R
      X1I=SQRTF(-DISC)/(2.0*A)
      X2I=-X1I
      RETURN
   60 X1R=-B/(2.0*A)
      X2R=X1R
      X1I=0.0
      X2I=0.0
      RETURN
   70 S=SQRTF(DISC)
      X1R=(-B+S)/(2.0*A)
      X2R=(-B-S)/(2.0*A)
      X1I=0.0
      X2I=0.0
      RETURN
      END(1,0,0,0,0,1,1,0,0,0,0,0,0,0,0)
```

```
N
  19  DUMP 1,40,6$A,B,C
  19  DUMP 1,1$COMMON DATA
  20  DUMP 1,3$A,B,C
  19  DUMP 1,20,5$PROGRAM DATA
N     SOLVE
  30  DUMP 1,40$IF (DISC+0.0) YES,NO,NO$DISC,A,B,C
  70  DUMP 1,40,7$S
  30  DUMP 1,40,5$DISC
  30  DUMP 1,40,6$COMMON DATA(1-3)
END OF DUMP REQUESTS
```

EXECUTION

```
        A              B              C            X1 REAL         X1 IMAG         X2 REAL         X2 IMAG

MAIN     EFN    20        COUNT     1

            A  4.0000000                B -3.0000000               C -5.0000000

SOLVE    EFN    30        COUNT     1

         DISC  89.000000

SOLVE    EFN    30        COUNT     1

   77457   -5.0000000       -3.0000000      4.0000000       0               0               0

SOLVE    EFN    70        COUNT     1

            S  9.4339812
   4.0000E 00    -3.0000E 00    -5.0000E 00     1.5542E 00                     -8.0425E-01

MAIN     EFN    19        COUNT     1

            A  4.0000000                B -3.0000000               C -5.0000000

MAIN     EFN    19        COUNT     1

   77453      0              -0.8042476      0              1.5542476      -5.0000000      -3.0000000
   77461     4.0000000        0              0              0               0               0

MAIN     EFN    19        COUNT     1

     235      2               1              0.0000271E-39  1.3559145E-20   0.0119221E-20   3.3103510E-24

MAIN     EFN    20        COUNT     2

            A  15.000000                B -11.000000               C -14.000000
   1.5000E 01    -1.1000E 01    -1.4000E 01     1.4000E 00                     -6.6667E-01

MAIN     EFN    20        COUNT     3

            A  5.0000000                B -4.0000000               C -3.0000000
   5.0000E 00    -4.0000E 00    -3.0000E 00     1.2718E 00                     -4.7178E-01
```

```
SOLVE     EFN     30        COUNT      4

          DISC -44.000000                       A  3.0000000                 B -2.0000000                 C  4.0000000
     3.0000E 00     -2.0000E 00     4.0000E 00     3.3333E-01     1.1055E 00     3.3333E-01     -1.1055E 00
     8.0000E 00      5.0000E 00     0.            -0.                           -6.2500E-01

SOLVE     EFN     30        COUNT      6

          DISC  77.000000
     1.0000E 00      9.0000E 00     1.0000E 00    -1.1252E-01                   -8.8875E 00

MAIN      EFN     19        COUNT      6

      235     2                     6             0.0000271E-39     1.3559146E-20     0.0119221E-20     3.3103510E-24

SOLVE     EFN     30        COUNT      7

          DISC -1.3500000                       A  4.0000000                 B  0.5000000                 C  0.1000000

SOLVE     EFN     30        COUNT      7

     77457     0.1000000        0.5000000         4.0000000          0              0               0
     4.0000E 00      5.0000E-01     1.0000E-01    -6.2500E-02     1.4524E-01    -6.2500E-02     -1.4524E-01

MAIN      EFN     19        COUNT      7

             A  4.0000000                       B  0.5000000                 C  0.1000000

SOLVE     EFN     30        COUNT      8

          DISC -47.000000                       A -3.0000000                 B  5.0000000                 C -6.0000000
    -3.0000E 00      5.0000E 00    -6.0000E 00     8.3333E-01    -1.1426E 00     8.3333E-01      1.1426E 00
     9.0000E 00     -1.2000E 01     4.0000E 00     6.6667E-01                    6.6667E-01
     1.0000E 00     -5.0000E 00     6.0000E 00     3.0000E 00                    2.0000E 00

SOLVE     EFN     30        COUNT     11

          DISC -59.000000                       A  3.0000000                 B -5.0000000                 C  7.0000000

SOLVE     EFN     30        COUNT     11

          DISC -59.000000
     3.0000E 00     -5.0000E 00     7.0000E 00     8.3333E-01     1.2802E 00     8.3333E-01     -1.2802E 00
```

```
MAIN     EFN    19       COUNT    11

    235     2                11                  0.0000271E-39     1.3559147E-20     0.0119221E-20      3.3103510E-24

SOLVE    EFN    30       COUNT    12

         DISC -31.000000                  A  5.0000000                    B -3.0000000                    C  2.0000000
    5.0000E 00    -3.0000E 00     2.0000E 00     3.0000E-01     5.5678E-01     3.0000E-01     -5.5678E-01

SOLVE    EFN    30       COUNT    13

         DISC -68.000000                  A  6.0000000                    B -2.0000000                    C  3.0000000

SOLVE    EFN    30       COUNT    13

    77457     3.0000000       -2.0000000       6.0000000          0                0               0
    6.0000E 00    -2.0000E 00     3.0000E 00     1.6667E-01     6.8718E-01     1.6667E-01     -6.8718E-01

MAIN     EFN    19       COUNT    13

             A  6.0000000                 B -2.0000000                  C  3.0000000
    8.0000E 00     0.             -7.0000E 00     9.3541E-01                    -9.3541E-01

SOLVE    EFN    70       COUNT     8

             S  41.291646
    9.0000E 00     2.5000E 01     -3.0000E 01     9.0509E-01                    -3.6829E 00

SOLVE    EFN    30       COUNT    16

         DISC  0
    4.0000E 00    -2.0000E 01     2.5000E 01     2.5000E 00                      2.5000E 00

MAIN     EFN    19       COUNT    16

    235     2                16                  0.0000271E-39     1.3559148E-20     0.0119221E-20      3.3103510E-24

SOLVE    EFN    30       COUNT    17

         DISC -16.000000                  A  5.0000000                    B -2.0000000                    C  1.0000000
    5.0000E 00    -2.0000E 00     1.0000E 00     2.0000E-01     4.0000E-01     2.0000E-01     -4.0000E-01
    5.0000E 00    -3.2000E 00     0.             6.0000E-01                    0.
```

7.4.0 Jul/65 Post Mortem Dumping

7.4.1 Jul/65 Introduction

The two subroutines, CRISIS and PØST, are used for obtaining a post-mortem dump in a FORTRAN II program. The main program and each subroutine should contain a CALL CRISIS statement which specifies the dump regions for the post-mortem dumping routine (PØST). Post-mortem dumps will be taken for all DEBUG or RECØM runs and for PRØDb runs which have been terminated by manual intervention or by a floating point overflow. During a PRØDb run PØST is executed only if a CALL PØST statement was used by the programmer. Writeups for CRISIS and PØST are included here.

CRISIS

PURPOSE: Specifies those regions in each program which the engineer wants printed out in his post-mortem dump.

USAGE: CALL CRISIS (SS, FS) where

SS = starting location of the dump region

FS = terminal location of the dump region

All arguments must appear in groups of two.

1. Storage Assignments

   a. Unsubscripted variables not in COMMON are assigned storage in reverse alpha-numerical order, i.e., from ZZZZZ to A1111.

   b. Subscripted variables, or arrays, not in COMMON are assigned storage first with respect to the dimension of the arrays, but in reverse order, i.e., 3, 2, 1, and secondly in reverse order of their appearance in the DIMENSION statement. Therefore the first named, lowest dimensioned array will occupy the highest memory location. This storage will immediately follow the storage of the unsubscripted variables.

   c. COMMON storage is assigned in reverse order of the naming of the variables and arrays on the COMMON card.

CRISIS (continued)

USAGE: (continued)

2. Notes

a. When dumping arrays in COMMON, using an array name for a starting location does not dump that array. Its storage lies ahead of the location of its name. Therefore if that array is desired in the dump, its name may be subscripted by its maximum size, R(20).

b. When EQUIVALENCE statements are used with names in COMMON, the storage assignments in COMMON are changed. With careful analysis of the storage map, correct CRISIS statements can be written.

c. No more than 50 CALL CRISIS statements may be executed in a single job. If the program attempts to do this the following message will be printed:

LIMIT ON SIZE OF CRISIS TABLE EXCEEDED

The job will not be terminated; however, all further CRISIS statements will be bypassed. CRISIS also sets up the conditions for initiating an IDUMP if such is necessary.

CRISIS (continued)

USAGE: (continued)

d. Output is described in the section on 7090 OUTPUT.

e. If CRISIS does not have at least two arguments, the entire statement is ignored.

3. EXAMPLE

```
COMMON R, S, T, A1, BETA, RI, G2, L, DN
DIMENSION L(10), DN (10, 20)
CALL CRISIS (G2, R, ZZ, AI, L, DN (10,20))
```

SOURCE: AL CRISIS (written by V. L. Sorensen)

REQUIRED SUBROUTINES:)SAVE(, DUMP, EXIT,)IDP(

POST

PURPOSE: Controls all selective post-mortem dumping from regions defined by CRISIS.

USAGE: CALL POST

Gives a post-mortem dump and terminates the job.

Routine is also called by

1. EXIT if job is in DEBUG or RECOM classification.
2. FPT if floating point overflow occurs.
3. PUC

If the specifications for the regions are in error, messages indicating this will be printed during the dumping process. These are listed below.

1. CORE SPANNED - OMIT DUMP

   This occurs when one of the limits of the dump region is in COMMON and the other is not.

2. DUMMY VARIABLE - DUMP MODIFIED

   This occurs when at least one of the limits of the dump is a dummy variable of the subprogram. The dummy variable is replaced by the highest storage location of the subroutine.

3. _____ BEGINNING _____ DUMP OMITTED

   This occurs when the CRISIS statement is incorrectly written, i.e., the arguments do not occur in groups of two.

7.4.3 (continued)
Jul/65

SOURCE: Written by V. L. Sorensen

REQUIRED SUBROUTINES: DUMP, CRISIS, UNIT(

7.5.0 Jul/65

Conversion of FORTRAN II to FORTRAN IV

While FORTRAN II jobs can be processed by the FORTRAN Monitor System in IBSYS, they will not operate under the IBJOB monitor. Incompatibility exists at both the binary and source deck levels. Since it may be desirable to prolong the operational period of a job by converting it to FORTRAN IV, a program exists to assist in this conversion called SIFT (Share Internal Fortran Translator). Details for usage of this program are in APPENDIX B.

7.6.0 Jul/65

FAP (Fortran Assembly Program)

8.0.0 Jul/65 GENERAL DCS OPERATING INFORMATION

8.1.0 Jul/65 <u>Job Priorities</u>

Every job which is processed on the 7040/7094 DCS system is given a job priority. This priority is used to position the job in the input queue. After a job has entered the card reader, DCMUP computes this priority using both the line count and the time estimates specified on the $JOB card. Short jobs are given a higher priority than long jobs, and the following method is used to determine this priority:

1) Equal weight is given to thousands of lines of output and minutes. A magic number (MN) is computed as follows:

$$MN = \text{minutes} + \frac{\text{lines}}{1000}$$

Example: Time estimate = 2.50 minutes
Line count estimate = 6050 lines

$$MN = 2.50 + \frac{6050}{1000}$$
$$= 2.50 + 6.050$$
$$= 8.550$$

The range of priorities for MN is:

MN = 0 - 4.999	Assigned priority 3
MN = 5 - 9.999	Assigned priority 2
MN = 10 - 14.999	Assigned priority 1
MN = 15 - over 15	Assigned priority 0

Priority 3 jobs are run first, 2 second, 1 third, and 0 last.

Examples:	Time Est.	Line Est.	MN	Priority
	5.00	5000	10.000	1
	10.00	5000	15.000	0
	2.00	2000	4.000	3
	2.00	5000	7.000	2
	2.55	625	3.175	3

2) If the time and/or line count estimate is not specified, the system limits will be inserted. The system limits are:

Production job:	10 minutes, 5000 lines	(priority 0)
Debug or Recom job:	5 minutes, 5000 lines	(priority 1)

3) If it is necessary to over-ride the above priorities in order to expedite special rush jobs, the DCS operator should be notified. In addition, the job must include a red EOF card obtained from S. M. Crandall at CAB.

Since time and line count estimates refer only to <u>execution</u> time and output, the programmer can make low estimates and always receive priority 3

8.1.0 Jul/65 (continued)

for jobs which are only to be compiled.

As described in an earlier section (6.2.3.3) the time and line count estimates on the $JØB card are used to terminate a job. The programmer should make certain that these estimates are correct. If a programmer estimates 30 minutes, the job <u>will</u> run 30 minutes unless stopped by calling EXIT or exceeding the line count estimates.

8.2.0 Jul/65 <u>Job Handling</u>

8.2.1 Jul/65 <u>Use of EAM Equipment</u>

Certain work can be done by EAM operators to assist the programmer with job preparation. The EAM equipment at CAB is in Room 129A, and any work to be done for the programmer should be placed in the appropriate box in Room 129. The services offered are:

1) Keypunching

2) Reproducing decks

3) Interpreting decks

4) Listing decks

5) Sorting decks

6) Re-sequencing decks

Two keypunches are available in Room 129 for programmers who have very small amounts of keypunching and wish to do it themselves.

8.2.2 Jul/65 <u>Keypunching Standards</u>

Whenever possible all jobs submitted for keypunching should adhere to certain standards. In this way they can be punched correctly and more quickly.

1) Use a standard form (available at CAB) for source decks, and follow field specifications. Extra blanks should be indicated by writing a lower case red "b" where they belong. The first card of every source deck should be a comments card containing its CAB number and the programmer's name.

2) Use standard forms for data cards (available in Room 129 and from stock).

3) Write legibly with a dark pencil and stay within the guide lines on the forms. Extra boxes on a line or extra lines on a form should not be added unless it is certain that they can be clearly understood and easily read.

8.2.2 (continued)
Jul/65

4) Specify card colors:

   a) Control cards, orange

   b) Debug cards, red

   c) FORTRAN IV decks, grey stripe FORTRAN cards

   d) FORTRAN II decks, brown stripe FORTRAN cards

   e) Rose-stripe FORTRAN cards are never used.

   f) Data cards, any color except those listed above.

5) Sequencing is recommended for both source and data decks. It is helpful to both the DCS operator and the programmer.

6) Distinguish carefully between characters that are easily confused:

Numeric	0	O
Alphabetic	O	Ø
Numeric	1	l
Alphabetic	I	I
Numeric	6	6
Alphabetic	G	G
Numeric	2	2
Alphabetic	Z	Ƶ

8.2.3 Job Submittal
Nov/65

After all the source decks and input data cards have been keypunched and checked, the decks are assembled for submittal to the 7040/7094 DCS. The programmer should briefly check:

1) That the appropriate EØF card is being used,

2) That all necessary control cards, source decks, and input cards are included, and

→ 3) **That $DECK cards are included for all necessary binary decks which are to be loaded from disk.**

8.2.3 (continued)
Nov/65

The job is then deposited in the appropriate box in Room 129 where it is picked up by the DCS operators for processing.

8.2.4 Job Pick-up
Jul/65

The programmer is notified when his job has been completed, and the job is placed in the appropriate box in Room 129. A decollator is available for the programmer's use if he wishes to remove the carbon paper from his listings.

8.3.0 Jul/65 **Job Accounting Procedures**

8.3.1 Jul/65 **CAB Number Assignment**

When a large group of research personnel uses the computing facility, it is inevitable that there be some duplication. However, this is always unfortunate and usually results in inefficient use of both time and manpower. In an attempt to minimize this, it is advisable that CAB have a general familiarity with all problems that are operating on the DC S. For administrative purposes it is essential that there be concentrated in one place a general knowledge of all computing projects. This also may enable CAB personnel to familiarize the user with other activities which parallel his interests. To assist the user in providing this information there is a form available at CAB which should be filled out for every new computing project. This form also serves as the request for the assignment of a CAB number. This form, a copy of which follows this section, includes the names of both the scientist and the programmer, the job order number, a brief description of the physical problem, a writeup on the mathematical techniques involved, and finally, it gives an estimate of computer time needed. This may be somewhat difficult to do but it must be emphasized that it is only by this technique that CAB gains any knowledge of what the Center's needs and requirements are. No attempts are made to enforce this estimate; it is only an indication to CAB.

8.3.1
Jul/65

Description of the Form

CAB NO. ____________ J.O. NO. ____________ DATE ________________

BRANCH ______________ ENGINEER ____________ PROGRAMMER __________

PHYSICAL DESCRIPTION ____

____

____

____

MATHEMATICAL TECHNIQUES ____

____

____

JOB MAGNITUDE ____

____

To give some assistance to the engineer in filling out this form there are below several categories for the physical description and for the mathematical techniques. If one of these is applicable to the project, the engineer may just use the code number to satisfy the description. If he feels that a little more detail would be beneficial to CAB personnel, he may elaborate. If the subjects do not cover his project, he will briefly describe

(continued)

it. More than one code number may be used if the problem is of that complexity.

Physical Problem

1. Aerodynamics, Aircraft
2. Aerodynamics, Missiles and Space Vehicles
3. Aircraft
4. Aircraft Safety and Noise
5. Atmospheric Entry
   a. drag devices and forces
   b. reentry maneuvers
6. Astronomy
7. Astrophysics
8. Behavioral Studies
   a. psychology
   b. personnel selection and training
   c. human engineering
9. Biomedicine
10. Biochemistry
11. Biology
12. Chemical Engineering
13. Chemistry, Inorganic
14. Chemistry, Organic
15. Chemistry, Physical
16. Cosmochemsitry
    a. chemistry of planetary and celestial bodies
    b. interstellar space
17. Communications and Sensing Equipment, Flight
18. Communications and Tracking Installations, Ground
19. Electronics
20. Fluid Mechanics
    a. hydrodynamics
    b. magnetic-fluid dynamics
21. Geophysics and Geodesy
    a. meterology
    b. ionosphere
    c. seismology
    d. solar streams
    e. planetary atmospheres
22. Guidance and Homing Systems
23. Launching Facilities and Operations

(continued)

24. Launching Dynamics
25. Materials, Engineering
26. Materials, Other
    a. lubrication and wear
    b. sealing compounds
    c. hydraulic fluids
    d. coolants
    e. shielding materials
    f. igniters
27. Mathematics
28. Missiles and Satellite Carriers
    a. weapons
    b. sounding rockets
    c. satellite launchers
29. Navigation and Navigation Equipment
30. Physics, Atomic and Molecular
    a. structures
    b. spectroscopy
    c. periodic system
31. Physics, Nuclear and Particle
    a. radiation
    b. nuclear reactions
    c. structures
    d. force fields
32. Physics, Solid State
    a. cryogenics
    b. crystallography
    c. semiconductors
    d. theories of elasticity
    e. plasticity
33. Physics, Theoretical
    a. classical mechanics, other than fluid mechanics
    b. magnetism
    c. optics
    d. acoustics
    e. wave and quantum mechanics
    f. thermal radiation
34. Piloting
35. Power Sources Supplementary
    a. auxiliary sources
    b. batteries
    c. solar and nuclear generators
36. Propellants

(continued)

37. Propulsion System Elements
    a. injectors
    b. nozzles
    c. heat exchangers
    d. pumps
38. Propulsion Systems, Air-Jet
    a. turbojets
    b. ramjets
    c. propeller systems
39. Propulsion Systems, Liquid-Fuel Rockets
40. Propulsion Systems, Solid-Fuel Rockets
41. Propulsion Systems, Electric
    a. ion jets
    b. plasma jets
42. Propulsion Systems, Nuclear
43. Propulsion Systems, Other
44. Propulsion Systems, Theory
45. Research and Development Facilities
46. Space Mechanics: orbital calculations and observations
47. Satellites
48. Space Vehicles
49. Simulators and Computers
50. Stability and Control
    a. aircraft
    b. missiles
    c. spacecraft
51. Stresses and Loads
    a. calculation methods
    b. structural tests
    c. fatigue
    d. vibration and flutter
    e. aeroelasticity
    f. stress analysis
52. Structures
53. Vehicle Performance
    a. specific flights
    b. observed performance

(continued)

Mathematical Techniques

1. Partial differential equations (please include order)
    a. elliptic
    b. parabolic
    c. hyperbolic
    d. mixture
2. Ordinary differential equations (please include number of equations in system, order and degree)
3. Integral Equations
4. Quadratures
5. Simultaneous Algebraic Equations
    a. Linear
    b. Non-Linear
    c. Matrix Manipulation
    d. Other
6. Roots of Equations
    a. polynomials
    b. non-polynomials
7. Interpolation
8. Curve Fitting
    a. Least Squares
    b. Tchebycheff
    c. Harmonic analysis
    d. Other

(continued)

9. Series

   a. Summation

   b. Manipulation

10. Special Functions

    a. Bessel Functions

    b. Legendre Polynomials

    c. Elliptic Integrals

    d. Others

11. Statistics, Probability

8.3.2 Jul/65

Machine Rental Charges

The rental of the 7040/7094 is charged to the J.O. which is used on the $JOB card for that particular job. 7094 time is computed by the use of the internal timer on the 7040 and measures the time the job is actually operating on the 7094. 7040 time is computed by taking the total lines of output and dividing it by 1200 to give minutes and hundredths of minutes used on the 7040. These two times are multiplied by a cost per minute figure determined each month from the actual rental paid on each machine. Since the rental figure varies each month, as well as the total utilization, this figure is not fixed. However, to permit calculation for work orders and job orders a few guidelines can be given.

1. Job order requests must include 7094 time and 7040 time. 7094 time can be estimated at $500 per hour, and 7040 time is approximately the same cost. However, to get an hourly figure, it can be assumed that 75,000 lines of output (this includes compilations) can be produced in one hour.

2. Work order requests require only 7094 time, and therefore the amount of money may be estimated at $500 per hour.

One final point, while the job order is charged for machine rental, the Center has a machine rental allocation independent of research and development funds, and all charges against job orders for machine rental are taken from this fund. This means that using the computing facility does not take money from research and development work.

8.3.3 Jul/65

Work Orders

Because machine time must be charged to the user's job order, it is essential that CAB have authorization to do a specified amount of computing. This is accomplished by submitting to CAB a work order stating the amount of 7040/7094 time required for a given project. This project is defined by naming the CAB numbers which are involved. Work will be done until this time is used, and at that time the engineer will be notified that the job is inoperable until further time is requested.

8.3.4 Jul/65

CAB Accounting Procedure

To provide the user with information on the amount of computer time he has used, and how much time remains on his work order, CAB does some preliminary accounting and produces bi-weekly reports giving these figures. This is posted in the DCS operation area for reference.

8.4.0 Jul/65 Magnetic Tape Usage

8.4.1 Jul/65 Introduction

Magnetic tape, because of its reliability and ease of handling, is presently the best means of storing large volumes of data. It is therefore necessary for users of computers to understand how information is stored on tape and how to program using tapes.

8.4.2 Jul/65 Definitions

To assist in this description, a few definitions are required.

1) Bit - One binary digit, either 0 or 1, stored in any one of 6 parallel channels or tracks on the tape.

2) Character - Six bits of information stored across the width of the tape encoded to represent alphanumeric characters and symbols.

3) Byte - Six bits of information stored across the width of the tape.

4) Word - Six characters or 36 bits of information (applies to 7040/7094).

5) Record - Continuous stream of words on tape.

6) End-of-Record Gap - 3/4" of blank tape between records produced by the output command writing the tape.

7) End-of-File - 3-3/4" gap of blank tape followed by a tape mark.

8) Tape Mark - Special character at the end of the 3-3/4" gap which is used to denote the end of a file.

9) File - A group of records separated by an end-of-file (EOF).

10) Parity - A bit in a seventh track on tape used to check the lateral validity of the recording in the six data tracks.

11) Binary - Tape which contains multiples of 36 bits of information and has odd parity, i.e., the total number of lateral ones in the seven tracks is odd. Tape written by a FORTRAN WRITE statement without a Format generates tape of this type.

12) BCD - Tape which contains multiples of 6 characters of information and has even parity, i.e., the total number of lateral ones in the seven tracks is even. Tape written by a FORTRAN WRITE statement which includes a Format generates tape of this type.

8.4.2 Jul/65 (continued)

13) Redundancy - Loss of parity i.e., on BCD tape there is odd parity.

14) Logical Record - A term applied to all words transferred by a single input/output statement in either FORTRAN or machine language. One logical record may be composed of one or more physical records.

15) Physical Records - The information contained between two end-of-record gaps. The number of words in this record may be limited by a system design - in FORTRAN, binary records are no longer than 256 words; in the DCS system tapes have 460 words in each physical record.

16) File Protection Ring - A plastic ring inserted in the back of a tape reel to permit writing on the tape.

17) Buffer - A block of storage which either receives input information before the program needs it or which holds output information prior to it being written on an output device.

18) Density - The number of characters or bytes written per inch of tape, currently 200, 556 or 800 characters per inch.

19) Load Point - The reflective marker at the beginning of the tape; recording of data starts beyond this marker.

20) Blocked Records - Physical records of 460 words which have been generated by transcribing physical records of arbitrary sizes which were generated on devices other than the DCS.

21) Deblocked Records - Physical records of arbitrary sizes which have been generated by transcribing 460 word physical records into those specified by the 7094.

8.4.3 Jul/65 Logical Tapes and Buffers

To permit simultaneous use of tapes, they are physically attached to one of several input/output channels. The user, however, specifies a tape by a logical tape number and the system assigns the appropriate physical unit. At Ames logical tape 5 is the system input device (SYSIN1) and logical tape 6 is the system output device (SYSOU1). This leaves six magnetic tapes with logical numbers 7 through 12 available for use by the programmer. Logical tape 3 may also be used for temporary storage of information which is not to be saved after execution is terminated. In FORTRAN IV the mode of the logical tape is controlled at job execution time. Logical tapes 7 through 12 are assigned modes by the system as follows:

7	BCD	8	BINARY
9	BCD	10	BINARY
11	BCD	12	BINARY

If, however, a programmer needs more BCD or binary tapes than the system permits, he may change the mode in the following manner:

From BCD to binary:

LOGICAL TAPE NO	SUBROUTINE REQUIRED
7	BIN07
9	BIN09
11	BIN11

From binary to BCD:

LOGICAL TAPE NO.	SUBROUTINE REQUIRED
8	BCD08
10	BCD10
12	BCD12

→ $DECK cards must be supplied for these subroutines but these subroutines are never referenced by the programmer. The system will make the necessary adjustments.

All the storage not used by the program is divided into many sections, or buffers, each of which can hold either one card image, one line of print, or one binary record. (Section 5.4.0). Since a card image contains 80 characters, 14 words (84 characters) are assigned to the buffer for the system input tape. The maximum length of all other BCD records is one line of print (22 words or 132 characters), and 22 word buffers are assigned to BCD tapes other than logical tape 5. Binary records written by FORTRAN, regardless of their logical length, are divided into physical records that do not exceed 256 words so 256 word buffers are provided for binary tapes. When a variable tape number is used, i.e., WRITE(N), the system does not know at load time what logical tape numbers will be used a exectuion time. Therefore, it must provide buffers to handle all tapes that are on the system and this results in an excessive use of core space and may over-run core. Several routines have been wirtten to eliminate buffers for logical tapes not used in the job. If the user wishes to remove logical tape numbers available for a given job, he may do so by adding → the appropriate $DECK cards to his deck makeup. The deckname for each tape which is not to be used follows:

LOGICAL TAPE NO.	BINARY DECK
3	DUM03
5	DUM05
6	DUM06
7	DUM07
8	DUM08
9	DUM09
10	DUM10
11	DUM11
12	DUM12

8.4.4 Jul/65

DCS Tape Information

8.4.4.1 Jul/65

Introduction

Since the DCS has disk as well as tapes available for storage, the system uses the disk whenever the programmer uses a tape unless the system has been directed otherwise. This information is transmitted to the system by the $SETUP card described in detail in Section 6.2.3. Such a system permits storage on the disk for the execution period of a job, but all information which must be retained beyond this time requires the use of a physical tape. Physical tapes can be placed in three categories:

1) Those generated by the DCS and used only by the DCS. This is the simplest since, although it does require a $SETUP card, it does not require blocking or deblocking and in general will be acceptable to the system.

2) Those produced by the DCS which serve as input to a device or computer other than the DCS.

3) Those prepared on a device other than the DCS which are used as input to the DCS. These last two classes can generate serious problems. The following discussion will outline some recommended procedures when the user has need of this type of usage.

8.4.4.2 Jul/65

Density

As previously stated, devices can write tape at different densities. Normal DCS operation processes tapes at 800 cpi. Therefore, if the external devices used absolutely require any other density, this information must be transmitted to the DCS operator. This is done by a note on the EOF card (Section 8.4.5). The only exception is a plot tape where the density is set automatically when the word PLOT is encountered on the $SETUP card.

8.4.4.3 Jul/65

Record Size

To optimize the use of the disk (Section 6.2.3.4) all tapes processed by the DCS have a fixed physical record size of 460 words. It is rather unusual to have an external device process this same fixed record size. Therefore, records on input tapes must be consolidated into physical records of this size (blocked) and output tapes must have physical records of this size separated into appropriate physical records as specified by the program (deblocked).

8.4.4.4 Jul/65

Deblocking of Output Tapes

The problems of deblocking for plotting, printing, punching and transmittal are usually not too severe although such activity may become time consuming.

8.4.4.5 Jul/65 **Blocking of Input Tapes**

The blocking of input tapes has several danger areas:

1) The original physical records are greater than 457 words.
2) The level of magnetization between the two tape transports may vary and cause excessive redundancy.
3) There may be several short (less than three words) binary records.
4) The total number of characters is not a multiple of 6.
5) The total number of bits is not a multiple of 36.

A brief discussion on each point:

1) Records larger than 457 words are blocked on the 7040 operating alone. This results in much slower service and a certain amount of inefficiency.
2) Serious redundancies almost force regenerating the original tape. If there is sufficient data that a redundant record may be accepted with no serious effect, the tape <u>may</u> be copied and the redundancy problem eliminated.
3) Short binary records are discarded as noise after one read of a redundancy occurs. <u>Avoid</u> records less than three words.
4) The final n characters ($n<6$) which form an incomplete word produce a redundant record and the tape cannot be blocked on the 7040 alone.
5) The final n bits ($n<36$) which form an incomplete word produce a redundant record and the tape cannot be blocked. It is obvious from the preceding discussion that if the user can exercise judicious control over the original method of generating the tape, many later problems can be avoided.

→ 8.4.4.6 Nov/65 **Maximum Number of Files**

No more than 80 files of information may be blocked or deblocked, if the disk is used.

8.4.5 Nov/65 **End-of-File Card**

Definitions for notations on the EOF card are:

1) Logical tape number - The number 7, 9, or 11 for a BCD tape, 8, 10, or 12 for a binary tape, which is used in the program to designate a particular tape to be read or written during the execution phase of the job. Logical tapes 5 and 6 and scratch tapes are not listed on the EOF card. A maximum of 6 tapes can be mounted for a given job.

2) Mount - If the library number of a tape appears in this column, that tape is to be mounted on a tape transport; if the word ASSIGN appears in this column, a tape is to be assigned a library number and mounted on a tape transport; if PLOT appears in this column, a plot tape is to be generated.

3) File Protect - Protect the tape designated under "Mount" from being written on during execution of this job by removing the file protection ring from the back of the tape reel.

8.4.5 Jul/65 (continued)

4) Print - This tape, written during execution of this job, is to be unblocked and printed off line. The number of files to be printed should be specified.

5) Punch - This tape, written during execution of this job, is to be unblocked and punched off line. The number of files to be punched, the approximate number of cards and the card color to be used should be specified.

8.4.6 Jul/65 Magnetic Tape Library

Since tape usage is extensive in this installation, certain rules for handling them must be established.

1) All tapes containing permanent information are given a library number when removed from the machine, with the exception of PLOT tapes. This number uniquely identifies this information for the operators. A tape number with a trailing U indicates the tape is unblocked; all others are blocked.

2) The programmer is given a white IBM card on which is punched the tape number, the programmer's name, the CAB number of the job and the date of creation.

3) An assigned tape is reserved for the programmer's exclusive use until he releases it. This is done by punching the release date on the IBM card, signing it and returning it to the DCS operators. If the tape is transmitted, the user should write on the card "Transmitted to _________ on ______________(date)". Tapes should be released as soon as possible.

8.4.7 Jul/65 Recommended Tape Usage Procedures

1) All auxiliary tapes used by a program should be rewound at the beginning of a program.

2) Tapes should be rewound during execution when they are no longer used. This saves time and money.

3) A tape must contain at least one end-of-file, if data has been written on it. More may be used to separate batches of information; in this case there must be a terminal end-of-file. Otherwise the tape runs away or erroneous data is processed.

4) ENDFIL permits a programmer to manipulate tapes when a job is terminated by no more input data on tape 5. This routine may be used to write terminal end-of-files.

5) Utility programs are available to obtain more information from tapes which are involved in debugging problems.

6) Tapes which contain data that are difficult or impossible to recreate should always have a copy for backup.

8.4.8 Jul/65

Subroutines

There are two classes of data which can be read or written by the DCS:

1) Those which can be handled by FORTRAN statements

2) Those which require special input/output routines. In addition, there is the need to manipulate multi-file tapes and to examine data which are currently on tape. A list of routines for these functions follows:

AL BNIN	Reads physical binary records of arbitrary length.
AL BDIN	Reads physical BCD records of arbitrary length.
AL NOWD	Reads physical records of BCD or binary tape of unknown and arbitrary length.
AL BIBC	Writes non-FORTRAN records in both BCD and binary modes.
AL NDID	Manipulates tapes containing non-FORTRAN records.
AL LOCATE	Positions multi-file tapes containing FORTRAN records.
AL TPDP	Dumps magnetic tape files (BCD or binary) and prints descriptive information of each physical record in the file being dumped.
AL TPSR	Counts the total number of physical records in a file and prints descriptive information for the first three records.
AL STAK	Copies BCD or binary files from a tape onto a master tape and prints a summary sheet. Mixed mode files may not be stacked.

8.5.0 Jul/65 Plotter Usage

8.5.1 Jul/65 Introduction

Two Electronic Associates, Inc. (EAI) Dataplotters are available to assist the programmer in preparing a visual representation of data generated by a computer program. These plots may range in complexity from a simple unlabelled "quick look" plot to a finished figure suitable for inclusion in a report. This section describes the plotting hardware, the available subroutines and their functions, and the instructions and restrictions for preparing plots.

8.5.2 Jul/65 Plotting Hardware

These two units are X-Y plotters which accept IBM-compatible magnetic tape as input. They have a 30" by 30" plotting table equipped with a pen and a 48-character symbol printer and an Ampex TM-4 tape transport which reads binary-coded decimal (BCD) tape at a density of 200 characters per inch. The plotters also have all the necessary hardware to convert the digital input to the analog voltages which are used for the actual plotting. All information to the plotters is in the form of six-character words, with one character being a command code which determines the action taken by the plotter. Records on a plot tape contain at least two such words. The coordinate of the points to be plotted consists of a sign, a command code, and four digits of data. Two modes of plotting are available, AUTO and FREE RUN.

In the AUTO mode each set of coordinates may be precisely plotted by either the pen or the symbol printer. With the pen, the plotting is done as a series of discrete <u>points</u> or as <u>lines</u> connecting successive points. With the printer, one of the following 48 symbols is printed at each point:

the numbers	0-9
the letters	A-Z
special characters	( ) + - * / = . , ⊡ ⊙ ◈

(The symbol is approximately centered about the actual point being plotted.)

In the FREE RUN mode, the plotter does "free-line" plotting, i.e., it plots with fairing, at speeds up to 4500 points per minute. To achieve acceptable results, the points to be plotted must be very closely spaced, and the function must have no sudden changes of magnitude. In this mode the tape is actually ahead of the plotter, and thus it is necessary to repeat the last point several times to insure that it is included in the plot.

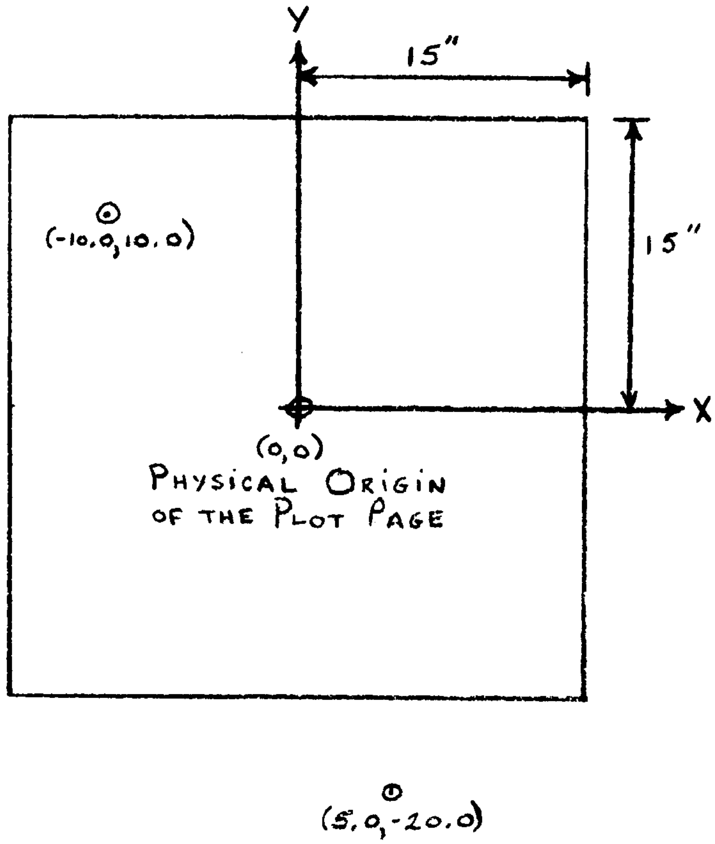
8.5.3 Jul/65 Description of Subroutines

The following information describes the plot page, as well as the subroutines which handle all transformations of data and the actual writing of the plot tape.

8.5.3.1 Jul/65

Plot Page

Many of the subroutines described below require origin information to be supplied by the programmer in terms of horizontal and vertical distances from the center of the plotting page. These distances are measured in signed inches as illustrated below.



8.5.3.2 Jul/65

Subroutines

AL PLOTWS	Plots the array Y versus the array X using either the pen or the symbol printer. Origin and scaling information are supplied by the programmer.
AL PLOT	Plots the array Y versus the array X using the pen. This routine should not be used in new programs since the FORTRAN IV version is a dummy routine which calls AL PLOTWS.
AL SCAL	Computes a scale factor on the basis of a given axis length and the maximum and minimum values to be plotted.

8.5.3.2 Jul/65 (continued)

AL MPLT — Provides, by means of a general purpose routine, automatic scaling and placement of the origin so that the maximum plotting area is 12" x 12". Both linear and logarithmic scales are available.

AL PSCA — Prints positive scale factors in the range 0.0001 to 9999.0 at one inch intervals along the X and/or Y axes.

AL SFAC — Prints scale factors at one inch intervals along the X and/or Y axes or the value of a single variable anywhere on the plotting page. Positive and negative numbers of any size may be printed in a choice of three modes - integer, decimal, or decimal with exponent.

AL PALP — Performs titling and annotation of plots using the pen. PALP can draw any of the 48 alphanumeric and special characters to the desired size at a desired location in either the X or the Y direction. If a character size of 1/10" is satisfactory, AL ALPP is more efficient.

AL ALPP — Performs titling and annotation of plots using the symbol printer. ALPP can print any of 48 alphanumeric and special characters at a desired location in either the X or the Y direction. The printed characters are approximately 1/10" high.

All of the above subroutines are available in FORTRAN IV. Only AL PLOTWS, AL PLOT, AL PSCA, and AL SCAL are available in FORTRAN II.

8.5.4 Jul/65 Preparation of Plots

8.5.4.1 Jul/65 Control Cards

1) A solid blue Plot card must be the first card of every plot job. It is described in detail below.

2) Each plot tape must be listed on the EOF card. Specify the logical tape number in the column LOGICAL TAPE NO. and write the word "PLOT" in the column MOUNT.

3) There must be a $SETUP card similar to the one in Example 5, Section 6.2.3.4 for each plot tape. Use the designations "PLOT1", "PLOT2", etc. under option 2. This is the only identification the DCS operators have to distinguish between plot tapes if there are more than one.

PLOT CARD

The Plot card serves two functions.

1) It informs the DCS operator that one or more of the tapes specified on a $SETUP card is a plot tape to be deblocked at 200 characters per inch.

2) It provides the Dataplotter operator with detailed instructions for plotting.

[illegible] NO. CAB	FOR	DATE	PHONE	MAIL

TAPE NO.	$SETUP OPTION	PAGE NO.	FILE NO.	PRINTER	PEN		PEN COLOR	FOR PEN LINE PLOT - MAX. DISTANCE BETWEEN POINTS	SPEC. INST. ATT.	LABEL
					POINT	LINE				

Each line of the Plot card describes the contents of <u>one</u> file of a plot tape since an EOF is necessary only if some action is required of the Dataplotter operator or if there is no more plot information. The use of unnecessary end-of-files is discouraged since each EOF stops the plotting operation.

1) TAPE NO. - Logical number of the plot tape.

2) $SETUP OPTION - Designation of the plot tape which appears in the Option 2 field of the $SETUP card, e.g. PLOT1, PLOT2, etc.

Items 1) and 2) provide a unique identification of the plot tape.

3) PAGE NO. - Number of the plotting page being described. The sheets of plotting paper are numbered sequentially.

4) FILE NO. - Number of the file on the plot tape.

5) PRINTER - Use of symbol printer is indicated by a check.

6) PEN-POINT - Use of pen to "point" plot is indicated by a check.

7) PEN-LINE - Use of pen to "line" plot is indicated by a check.

Only <u>one</u> of items 6) or 7) may be checked for each file since manual intervention is required to change from the "point" to the "line" mode.

8) PEN COLOR - Color for the pen, if used (red, green, black, or blue). Red is the recommended color.

9) MAX. DISTANCE BETWEEN POINTS - Maximum distance in inches between two adjacent points (Section 8.5.4.2).

8.5.4.2 Jul/65

Maximum Point Displacement Control

When plotting lines with the pen, the plotting speed must be set manually. It is inversely proportional to the "maximum point displacement", i.e., the maximum distance between adjacent points. Once this speed is selected, via the "Maximum Point Displacement" switch on the plotter console, manual interruption of the plotting is required to change it. Consequently, data in the same file with widely differing maximum distances between adjacent points cannot be plotted efficiently. For example, when plotting a curve and its axes, the maximum displacement between adjacent points on the curve may be approximately 1/4", whereas for the axes it may be 2 inches. If all this information is in the same file, then the entire plot must be done at a speed which permits accurate plotting of the points which are 2 inches apart. However, if there are two files, the curve itself could be plotted at a much higher speed.

A space has been provided on the Plot card so that the programmer may specify the maximum point displacement in inches, if it is known, for all line plots. If this information is not specified, the Dataplotter operator must run through the plot with the pen up to determine approximately how fast it can be plotted. For plots prepared with the subroutine AL MPLT, the maximum point displacement is always 1/4 inch. The necessary logic to achieve this has been included in the subroutine.

8.5.4.3 Jul/65

Restrictions

1) Programmer Responsibilities
   a) Plot tapes should be rewound before the first reference to any plotting subroutine. This insures that the tape is positioned at the very beginning, i.e., at "load point". (It is recommended that the programmer rewind the tape at the end of the job.)
   b) An end-of-file must be written at the end of a plot tape and at any point where manual intervention of the plotting by the Dataplotter operator is required. Manual intervention is required to change the paper, the ink color, or the setting of a switch on the plotter console (e.g., AUTO or FREE RUN, LINE or POINT, or MAX. POINT DISPLACEMENT).

2) The actual area available for plotting is 28 inches in the X direction and 26½ inches in the Y direction, centered about the physical center of the 30" x 30" paper. The reduction in the plotting surface is necessary to prevent collisions between the pen and the printer, and the printer and the sides of the plotting table.

3) The classification of the $JOB card must be either RECOM or PRODb since no plotting is done for DEBUG runs.

4) Switching between the pen and the symbol printer should be minimized since it is very time consuming.

5) If there is any overlapping of symbols and lines, it is recommended that the symbols be plotted first to avoid smearing the ink left by the pen.

8.5.4.3 (continued)
Nov/65

6) Since the accuracy of the plotters is 0.015 inches, it is unnecessary to plot points which will be transformed into plotter coordinates which are closer than this resolution.

7) For plotter operation purposes it is desirable that all plot tapes have an end-of-file written at the beginning of the tape (be sure to include this file on the $SETUP card).

8.6.0 Jul/65

Subroutine Abstract Usage

Although individual problems require separate computer programs, it is frequently possible to use general purpose subroutines to accomplish some of the mathematical, input/output, or other processes which are involved. The Ames Program Library provides the user with a variety of these subroutines. A portion of this library consists of programs which have been written by other IBM users and distributed through SHARE, the organization of 704X and 709X users. The remainder is composed of programs which have originated at Ames. The library provides documentation on all of these routines, and many of the actual programs are available.

All programs are categorized according to the SHARE subject codes as listed in the Ames Subroutine Manual and are identified by the author's SHARE installation code and a program name. A functional abstract of each program has been made, and these are listed alphabetically by installation code and program name. In addition, certain words from the abstract have been selected as "key words" and have been combined into a description of the program. Two additional listings are provided to enable the user to locate a pertinent routine.

1) Key-Word In Context (KWIC)
   All descriptions are permuted on the key-words and then alphabetically listed. Installation code and program name are to the right of the description and provide reference to the appropriate abstract.
2) Subject Code
   All routines are sorted by their primary subject code and are listed alphabetically within each group by installation code and program name to provide a reference to the appropriate abstract.

In the listing of the abstracts, the abstract itself is preceded by the following information:

1) Subject codes
2) Machine
3) Date of coding
4) Author
5) Available documentation
6) Available materials (decks)
7) SHARE distribution number

Availability of items 5 and 6 falls into three classes:

1) Must be ordered from New York which takes approximately three weeks.
2) Must be obtained from Ames Master Tape which takes two or three days
3) Is available for immediate use. These routines are denoted by a trailing "A" on the distribution number or by the word "Ames" in the KWIC index.

The abstract listings as well as the indexes are available in Rooms 101 and 129 of the Data Reduction Center.

9.0.0 Jul/65 SPECIALIZED INPUT/OUTPUT INFORMATION

9.1.0 Jul/65 Paper Tape Usage

9.1.1 Jul/65 Experimental Design

Scientists who are using paper tape for recording data from their experiments should adhere to certain paper tape standards whenever possible. In addition it is advisable that projected experiments be discussed with CAB while they are still in the design stage. Part of the function of CAB is to advise on tape format, the availability and validity of mathematical and statistical techniques, and the machine configurations available to produce and reduce data.

The design of an experiment should make provisions for:

1) Identification
The first record of each file of paper tape should contain sufficient information to uniquely identify the data.

2) Using standard BCD character codes

3) Auxiliary recording equipment
The data should be recorded on some auxiliary device running in parallel with the paper tape punch to insure that data is not lost through equipment malfunction.

4) Extra data
If the experiment allows, more data than is required should be taken to replace erroneous data.

5) Uniform record length

6) Reuseability of data
If there is a possibility that the data will be used for further analysis, this possibility should be considered in choosing a tape format and discussed with CAB in the original program design.

7) Check tapes
If possible, a check run should be devised to check all phases of the experiment before data acquisition begins.

9.1.2 Jul/65 Paper Tape Standards

1. Equipment
A large percentage of the difficulties encountered in reading paper tape is caused by data acquisition equipment malfunction. An adequate maintenance schedule must be maintained.
Mr. Harmon of the Electronic Instrument Branch will specify an

equipment check to be run each time the equipment is turned on for use, and he should be informed of any malfunction at the time it occurs. Section 9.1.4 presents the acceptable tolerances for punching paper tape.

2. Physical Characteristics
The physical characteristics listed below are to be used. Any exceptions or deviations should first be discussed with CAB personnel.

1) Paper tape
   a. Black or pink oiled tape.
   b. Seven level or eight level.
   c. Loaded on punch supply reel so manufacturer's label can be read as tape passes through punch or reader.

2) Physical format of punched tape
   a. Length of tape
      1) Maximum length that can be mounted on the paper tape reader is one complete 8 inch diameter reel.
      2) There is no minimum length but the tape should contain all data that will be reduced. Short lengths of tape increase the computer operator's workload and must be avoided.
   b. Leader
      A three-foot leader containing only sprocket holes or sprocket holes and a unique character must be punched at the beginning and end of every reel.
   c. Characters
      1) A maximum of 60 characters may be used in one experiment. This count includes end-of-file characters, end-of-record characters, end-of-word characters, and leader character.
   d. Words
      1) An end-of-word character is optional but may be used to segment a record.
      2) An end-of-word mark, if used, must be unique.
   e. Records
      1) A record may be no longer than 1200 characters.
      2) A record must be terminated by a unique end-of-record character.
      3) No more than 5 different end-of-record characters may be used in one experiment.
   f. Files
      1) A file must be terminated by a unique end-of-file character immediately following the last end-of-record character.
      2) No more than 5 different end-of-file characters may be used in one experiment.
   g. Parity
      1) Parity is optional but is recommended for error detection.
      2) Even, odd, or no parity may be used.
   h. Tape quality
      Tape that has been spliced or otherwise mutilated cannot be read.

9.1.3
Jul/65

Dictionary

BCD The system of encoding decimal numbers by representing each digit in a decimal number by its binary equivalent. Alphabetic characters are encoded by using the numeric bits plus the remainder of the other three (seven level) or four bits (eight level tape).

Bit The abbreviation for a binary digit. On an eight level tape there is the possibility of eight bits located laterally across the tape to form a logical character.

Character A combination of bits. A character may be represented by a combination of any of the bits located laterally across the tape.

Word A group of characters that form a piece of data.

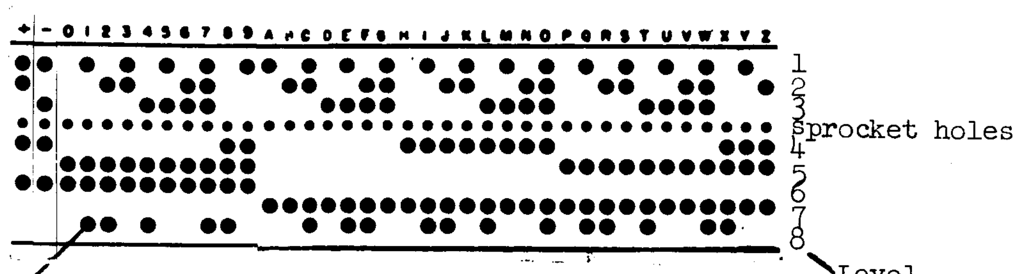
Record A group of words that forms a data set. A record shall always be immediately followed by an end-of-record character.

File A sequential set of data records that pertains to one function, test period, or other logical grouping. A file shall always have an end-of-file character immediately following the last end-of-record character.

Parity A bit that is punched in a predesignated level so that the total number of bits punched in a logical character is always either odd or even. (The parity bit allows the computer program to detect mispunched or misread characters, but not to correct them).

Tape A reel containing one or more files.

Standard Numeric and Alphabetic Characters



Parity bit (even parity). Parity bit may be in some other level if so designated.

9.1.4 Jul/65

Proposed American Standard - One-inch Perforated Paper Tape for Information Interchange

1. Scope

   a. This standard covers the physical dimensions of the paper tape and its perforations.

   b. This standard is for perforated paper tape with fully punched round holes.

2. Standard One-inch Perforated Paper Tape

   a. The unpunched tape shall have an overall width of 1.000 inch plus or minus .003 inch after the tape has been conditioned to within 69.5°F to 76.5°F and 48% to 52% relative humidity for a period of 24 hours.

   b. The unpunched paper tape shall have a thickness of .004 inch plus or minus .0003 inch after the tape has been conditioned to within 69.5°F to 76.5°F and 48% to 52% relative humidity for a period of 24 hours.

   c. The tape shall be used for recording up to eight levels of information across the tape.

   d. The longitudinal center line of the feed holes shall be parallel with the guided edge of the tape and located .392 inch plus or minus .003 inch from the guided (3-track) edge.

   e. The feed holes in the tape shall be fully punched and round with a diameter of .046 inch plus .002 minus .001 inch.

   f. The code holes in the tape shall be fully punched and round with a diameter of .072 inch plus or minus .002 inch.

   g. All holes punched in the tape shall nominally center on the true intersection of longitudinal and perpendicular transverse center lines spaced .100 inch apart. Tolerances on locations of code holes in any one transverse row, relative to the center line of the feed hole in that row, shall be plus or minus .002 inch in the transverse direction and plus or minus .003 inch in the longitudinal direction. Tolerance on distance between centers of adjacent feed holes shall be plus or minus .002 inch. Accumulated longitudinal error between centers of feed holes shall not exceed plus or minus .015 inch within spans of .8 to 6 inches.

   h. The dimensions and tolerances given in Paragraphs 2.d through 2.g for the holes and their spacing are to be met at the time of punching or with the tape conditioned to the same environmental conditions as those which existed at the time of punching.

   i. The tape layout specified above is illustrated for reference in Figure 1.

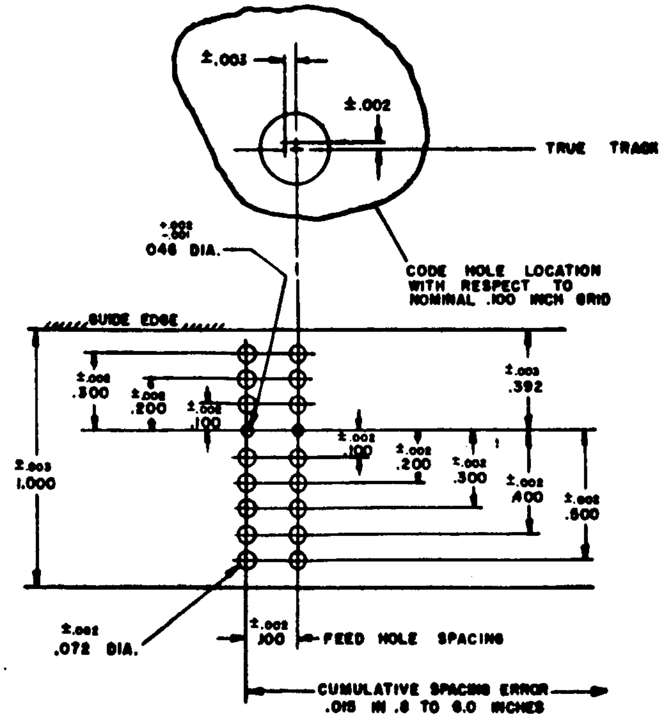


FIG. 1

9.1.5 Jul/65 Miscellaneous Notes

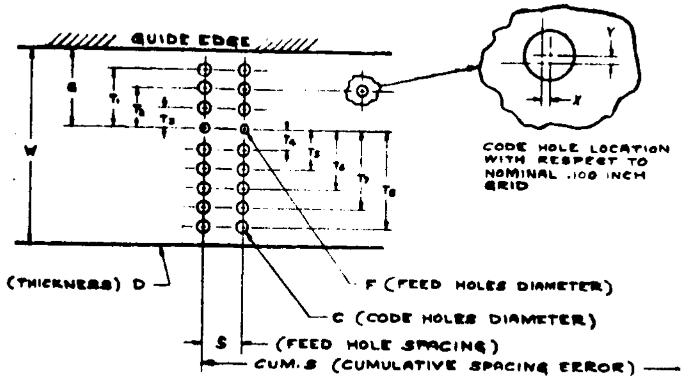
Note 1. Environmental Considerations

Paper (as commonly produced and used for perforated tapes) is inherently subject to some change in dimension with changes in environmental factors, particularly humidity. Interchanging of perforated paper tape between equipments in dissimilar environments may disclose dimensional variance beyond the limits of this standard.

Note 2. Tape Materials Other Than Paper

The physical dimensions specified in this standard, with the exception of tape thickness (Paragraph 2.b of section 9.1.4), also apply where materials other than paper are used.

COMPARISON OF DIMENSIONAL STANDARDS
ONE-INCH PERFORATED TAPE



Dimension	EIA 11/16"	EIA 1" RS-227	AIEE 1" No. 89	Proposed American Std	Difference
W	-	1.000 ±.003	1.000 +.005 -.004	1.000 ±.003	Tolerances
G	-	.392 ±.003	.392 ±.003	.392 ±.003	-
F	.046 +.002 -.001	.046 +.002 -.001	.046 +.002 -.001	.046 +.002 -.001	-
C	.072 +.001 -.002	.072 +.001 -.002	.071 ±.002	.072 ±.002	Tolerances
T_1	-	.300 ±.002	.300 ±.002	.300 ±.002	-
T_2	.200 ±.002	.200 ±.002	.200 ±.002	.200 ± .002	-
T_3	.100 ±.002	.100 ±.002	.100 ±.002	.100 ±.002	-
T_4	.100 ±.002	.100 ±.002	.100 ±.002	.100 ±.002	-
T_5	.200 ±.002	.200 ±.002	.200 ±.002	.200 ±.002	-
T_6	.300 ±.002	.300 ±.002	.300 ±.002	.300 ±.002	-
T_7	-	.400 ±.002	.400 ±.002	.400 ±.002	-
T_8	-	.500 ±.002	.500 ±.002	.500 ±.002	-
X	±.003	±.003	±.003	±.003	-
Y	±.002	±.002	±.002	±.002	-
D	.004 ±.0003	.004 ±.0003	.004 ±.0003	.004 ±.0003	-
S	0.100 ±.002	0.100 ±.001	0.100 ±.0015	0.100 ±.002	Tolerances
Cum. S.	(.009 in. 0.9" to 6.0")	(.009 in 0.9" to 6.0")	(.015 in 0" to 6.0")	(.015 in .8" to 6.0")	Dimension

FIG. 2

9.1.5 (continued)
Jul/65

Note 3. Discussion

The existing or proposed standards which were reviewed and the resolution of their differences are shown in Figure 2.

In general, the variations between existing standards were in minor tolerance differences. A more substantial difference existed with respect to allowable cumulative error in pitch (.009 inch vs. .015 inch in up to 6 inches).

The cumulative pitch error variation was resolved in favor of the less stringent tolerance because that tolerance was satisfactory for almost all existing equipment, and it was not felt desirable to burden information interchange with the exceptional requirements of a small number of special devices.

9.1.6 Jul/65 Program for Punching Paper Tape

A self-contained program for the Honeywell 200, IBMPT2, is available which reads binary record information from a ½" magnetic tape and then punches this information on a paper tape. The records of the magnetic tape must be 556 cpi density. Maximum record length is 500 paper tape characters. A detailed write-up describing the usage of this program is available from the Honeywell 200 Library File.

9.1.7 Jul/65 Program for Reading Paper Tape

A self-contained program, PTMIBM2, for reading data from a paper tape and then writing this data on a ½" magnetic tape is also available. The program operates on the Honeywell 200, and the magnetic tape produced can be processed on the 7040/7094 DCS system if desired. A complete write-up of this program is available from the Honeywell 200 Library File.

9.2.0
Jul/65

Analog to Digital Conversion of Magnetic Tapes

Equipment is available to digitize data recorded on magnetic tape. Certain operational procedures have been established to enable efficient processing of the digital data. Because of the relative freedom with which data can be recorded with analog techniques, some decisions on the format and content of the digital tape must be made before experiments are started.

Before digitizing any of the data, a discussion with personnel at CAB about the computations to be made with the data is recommended. To adequately define the problem, a written formulation of the computations to be performed on the data, quantities to be output, order of output, and any other information significant to the processing of the data are required. This information and discussion will define any problem areas and will allow more time for required programming.

It is currently planned to use the SDS920 computer, operated by the Guidance and Control Systems Branch, for converting all analog magnetic tapes to digital magnetic tapes (IBM compatible). In the conversion system there are certain limitations as follows:

1) There are 13 binary bits plus sign to represent the analog signal. This corresponds to a decimal number of $\pm$ 8192. These bits will be arranged on the digital tape in the following manner:

```
± x x ± x x
x x x x x x
x x 0 x x 0
x x 0 x x 0
x x 0 x x 0
x x 0 x x 0
```

SDS 920 word

36 bit IBM word

2) The maximum number of times an analog tape may be sampled, without special provision in writing the analog tape, is 10,000.

3) Multiple analog tracks <u>cannot</u> be sampled simultaneously. If this is required for time dependent correlation between tracks on the analog tape, consultation with the C & A and the GCS Branches will be necessary.

4) The first file on the digital tape will consist of three 6 character (six 3 character SDS920 words) BCD alphanumeric words for identification of the job. This identification will only be written once on each digital tape.

5) The first record of all subsequent files will consist of at least three and no more than six 6 character BCD alphanumeric words for identification of the data that follows.

6) All subsequent records within the file will be data records and will consist of five hundred 18 bit words (only 13 bits plus sign of the 18 bits will be used for the data sampling). This will result in a maximum of 20 data records per file.

7) In pre-test planning, adequate provision must be made so that the non-data sections of the analog tape will not be digitized on the SDS 920 computer.

If the above limitations prove to be too restrictive, consultation with the Computation & Analysis Branch will hopefully provide a satisfactory solution.

Appendix A

DIRECTORY

1.0.0 DIRECTORY
Nov/65

1.1.0 Physical Plant
Nov/65

All personnel and equipment related to the DCS are located in the left wing and center section of the first floor of the Data Reduction Building. All computer rooms are accessed through an operations office, which is the office at which work for the computer will be deposited and returned. On the second floor there is located a classroom which will be used for all scheduled classes and for general discussions when they are announced. The following is a directory for all rooms not used as office space:

	Description	Number
→	Computer and EAM Supervisor	127a
→	EAM Room	115
	Theoretical	117
	Theoretical 1401	117a
	DCS Operations	129
	Peripheral Equipment	129a
	DCS Computer Room	127a
	H-800 Operations	133
	H-800 Computer Room	133a

1.2.0 Personnel
July/66

COMPUTATION AND ANALYSIS BRANCH

Data Reduction Building, N-233-7

Chief:	W. A. Mersman	Room 105 - Ext. 2333
Assistant Chief:	S. M. Crandall	Room 132 - Ext. 2452
Staff Specialist:	M. K. Chartz	Room 104 - Ext. 2349
Secretary:		Room 103 - Ext. 2208

Mathematical Analysis Section

Head: P. F. Byrd		Room 108 - Ext. 2443
	Card, D. H.	Room 120 - Ext. 2448
	Jeske, J. A.	*Room B1 - Ext. 2877
	Lee, K. L.	*Room B1 - Ext. 2877
	Resnikoff, M. M.	Room 100 - Ext. 2380

Theoretical Computation Section

Head:		
	Finch, W. P.	Room 100 - Ext. 2380
	Gaspar, J. C.	Room 109 - Ext. 2444
	James, N. A.	Room 118 - Ext. 2447
	King, M. L.	Room 114 - Ext. 2895
	Sorensen, V. L.	Room 118 - Ext. 2447
	Tashjian, H.	Room 112 - Ext. 2446

Experimental Data Reduction Section

Head: N. K. Delany		Room 106 - Ext. 2427
	Covert, M.	Room 109 - Ext. 2444
	Crawford, W. L.	Room 128 - Ext. 2504
	Kelton, H. A.	Room 124 - Ext. 2507
	Louie, P. T.	Room 124 - Ext. 2507
	Luke, R. C.	Room 128 - Ext. 2504
	Phifer, J. L.	Room 110 - Ext. 2445
	Sutton, R. E.	Room 122 - Ext. 2674
	Yamada, S. S.	Room 110 - Ext. 2445

Systems Programming and Operation Section

Head: S. M. Crandall		Room 132 - Ext. 2452
	Kirkeby M. N.	Room 126 - Ext.
	Devine, J. P.	Room 126 - Ext.
	Sheaffer, R. F.	Room 130 - Ext. 2449
	Wentz, M. K.	Room 114 - Ext. 2895

1.2.0 Personnel - (continued)
July/66

H-800 Operators

Ginsberg, R. A.	Room 133 - Ext. 2518
Golding, S.	Room 133 - Ext. 2518
McCarthy, T. E.	Room 133 - Ext. 2518
Sakamoto, R.	Room 133 - Ext. 2518
Streich, H. C.	Room 133 - Ext. 2518
Holzer, G. A.	Room 133 - Ext. 2518

IBM Operators

Davenport, D. M.	Room 127 - Ext. 2453
Gonzales, T. G.	Room 127 - Ext. 2453
Loop, D.	Room 127 - Ext. 2453
Pfeiffer, G. M.	Room 127 - Ext. 2453
Rippy, R. L.	Room 127 - Ext. 2453

EAM Operators

Alfaro, L. S.	Room 129 - Ext. 2453
Baer, M. B.	Room 129 - Ext. 2453
Mitchell, M. A.	Room 129 - Ext. 2453

Tab Room, Unitary Building

Settles, E. D.	Ext. 2209

* B1: Basement, Room 1

Appendix B

SIFT

Conversion of a FORTRAN II Program to a FORTRAN IV Program

1.0.0 Jul/65 CONVERSION OF A FORTRAN II PROGRAM TO A FORTRAN IV PROGRAM

1.1.0 Jul/65 Introduction

There are two basic reasons why FORTRAN II source decks must be recompiled before they can be executed under the IBJOB monitor. The first is that the text and format of the binary object decks are different, and the second is that source language incompatibilities exist between FORTRAN II and FORTRAN IV. In general, FORTRAN II source decks require modification before they can be compiled under the IBJOB monitor. Some of the modifications must be made manually, but most can be made by use of the FORTRAN II conversion program SIFT. The usage of SIFT is described below, and its powers and limitations are discussed in detail.

2.0.0 Nov/65 USAGE OF SIFT (SHARE INTERNAL FORTRAN TRANSLATOR)

→ SIFT is a FORTRAN IV program which uses the decks to be translated as data. The deck makeup for a SIFT run is special and is as follows:

EOF card (use green stripe card)

$JOB card (standard) — Note the absence of the $IBJOB card

$DATA

____

______________ — any number of FORTRAN II decks

____

On the EOF card the binary deck SIFT4 must be requested under "Binary Deck Makeup".

The *LABEL and *SYMBOL TABLE cards are ignored by SIFT, and they may be removed or retained as desired.

The output from a SIFT run is:

1) Listing of FORTRAN II input program.
2) Listing of the resulting FORTRAN IV program with notations which indicate the changes that have been inserted.
3) New FORTRAN IV symbolic deck.

No binary decks are produced by SIFT. A resulting FORTRAN IV deck must be compiled under the IBJOB monitor to obtain a binary deck.

The FORTRAN IV deck produced by a SIFT run is sequenced in columns 77 - 79. The contents of columns 2 - 5 of the first comments card of each FORTRAN II deck are reproduced as identification in columns 73 - 76 of the corresponding FORTRAN IV deck. If the first card of the FORTRAN II deck is not a comments card, then columns 73 - 76 of the resulting FORTRAN IV deck are left blank.

2.1.0 Jul/65

Discussion

The language and usage differences between FORTRAN II and FORTRAN IV are discussed in detail in Appendix C of the publication

IBM 7090/7094 Programming Systems
FORTRAN IV Language
Form C28-6274-3

and need not be repeated here. SIFT will treat in a correct manner all of the incompatibilities described there, with some reservations with regard to double-precision and complex arithmetic (see discussion to follow on restrictions). In addition SIFT will:

1) Change the G-format to the E-format and

2) Change the
```
    READ 12, list
```
statement as well as the
```
    READ INPUT TAPE 5, 12, list
```
statement to
```
    READ (5,12) list.
```
The FORTRAN II statement
```
    READ 12, list
```
must have a blank separating the READ and the format number.

FORTRAN II Boolean Algebra statements are properly converted by SIFT in the following way: Boolean constants are introduced by use of FORTRAN IV DATA statements, and the logic is performed by use of FORTRAN IV library subroutines AND, OR, COMPL, BOOL.

It is to be noted here that SIFT introduces new symbols as required in order to effect the conversion. For example, the FORTRAN II statement A = 6HBCDEFG is changed to
```
    DATA Q000CT/6HBCDEFG/
    A = Q000CT
```
where the symbol Q000CT has been inserted by SIFT.

→ 2.2.0 Nov/65

RESTRICTIONS

1) SIFT does not alter or replace the CALL CRISIS statements. These must be removed manually.

2) SIFT does not change logical tape numbers to conform to FORTRAN IV usage.

3) SIFT does not add any coding to set variable or array locations to zero initially. The programmer cannot expect core storage to be set to zero initially in FORTRAN IV usage. Unused memory contains a special non-zero code word.

(continued)

4) SIFT will convert double-precision and complex variables, arrays, statements, etc., consistent with the general discussion of Appendix C of the publication cited above. Such programs frequently fail to execute properly, due to logic difficulties that arise because of differences in the way in which the two parts of double-precision and complex variables are stored in arrays. It is frequently more satisfactory to rewrite these programs directly in FORTRAN IV rather than to try to make a SIFT-produced version execute correctly.

5) Programming to accommodate tape manipulation for I-dumping and, in particular, references to TØTAPE, TØCØRE, and KEEP are not deleted and must be treated manually.

6) In using SIFT to convert programs received by transmittal, such statements as CALL SSWTCH (i,j) and PRINT n, list, which are not permitted on this system, must be eliminated or altered manually.

ABSTRACTS	8.6.0	
ACCOUNTING		
SEE COMPUTER TIME		
ACCOUNTING RECORD	6.5.0	8.3.0
ANALOG		
COMPUTER ON LINE	4.5.0	
CONVERSION TO DIGITAL	9.2.0	
ARROWS	3.0.0	
BUFFER	5.4.0	8.4.2
	8.4.3	
BUFFER SIZE	6.2.3.4	8.4.3
CAB NUMBER	6.2.3.10	6.2.3.2
	6.2.3.3	8.3.1
CARDS		
$JOB	6.2.3.3	7.2.3.3
$SETUP	6.2.3.4	
COLORS	6.2.2	7.2.2
	8.2.2	
CONTROL (II)	7.2.3	
CONTROL (IV)	6.2.3	
DEBUG REQUEST	6.3.4	6.3.5
EOF (II)	7.2.3.2	8.4.5
EOF (IV)	6.2.3.2	8.4.5
PLOT	6.2.3.1	7.2.3.1
	8.5.4.1	
PRIORITY	8.1.0	
CHAIN TAPES	5.7.4	
COBOL	6.8.0	
COMMON, BLANK (//)	6.5.0	
COMMON, LABELLED	6.3.5.2	6.5.0
COMPUTER TIME		
ESTIMATED EXECUTION	6.2.3.3	8.1.0
J. O. CHARGES	8.3.2	
MAXIMUM EXECUTION	6.2.3.3	8.1.0
SYSTEM LIMIT	6.2.3.3	8.1.0
CONTROL CARD OPTIONS		
SEE OPTIONS,CONTROL CARD		
CONTROL CARDS		
SEE FORTRAN IV AND II		
COST, MACHINE		
JOB ORDER	8.3.2	

7040	8.3.2	
7094	8.3.2	
CRISIS	7.4.2	
DCMUP	5.2.0	
DCS		
EQUIPMENT	4.1.0	
OPERATING DETAIL	8.0.0	
OPERATING SYSTEM	5.1.0	
RECORD FORMAT	6.2.3.4	8.4.4.3
TAPE INFORMATION	8.4.4	
DD		
SEE OPTIONS,CONTROL CARD		
DEBUGGING (II)		
*DEBUG	7.2.3.9	
*SYMBOL TABLE	7.2.3.8	7.3.2
DEBUG MACROS	7.3.4	
DUMPS		
DYNAMIC	7.3.0	
POST-MORTEM	7.4.0	
EXAMPLES	7.3.6	
FORMAT	7.3.4.1	7.3.4.2
	7.3.5.2	
NOTES	7.3.5	
PLACEMENT OF MACROS	7.3.5.4	
STORAGE	7.3.5.5	
DEBUGGING (IV)		
CONTROL CARDS		
$IBDBL	6.3.3	
*DEBUG	6.2.3.7	6.3.3
*DEND	6.2.3.9	6.3.3
*ETC	6.2.3.8	6.3.3
DEBUG PACKET		
NOTES	6.3.7	
PLACEMENT	6.2.1	6.3.3
	6.3.7	
DEBUG REQUESTS		
EXAMPLES	6.3.4	6.3.8.1
	6.3.8.2	
EXECUTION	6.3.7	
FORMAT	6.3.6	
FREQUENCY	6.3.5.1	6.3.5.1
	6.3.7	
FUNCTION	6.3.4	
PLACEMENT	6.2.1	6.3.3

	6.3.7	
RESTRICTIONS	6.3.6	6.3.7
STATEMENTS	6.3.5	6.3.5.2
TABLE	6.3.5.1	
DECK	5.5.0	
BINARY	6.2.3.10	7.2.1
	7.2.3.7	8.2.3
ADD TO DISK	5.6.5	
LABEL OF	6.2.3.10	7.2.3.7
MAINTENANCE	5.6.1	
PURGE FROM DISK	5.6.4	
REQUEST FROM DISK	6.2.3.11	7.2.3.11
	8.2.3.11	
USE FROM DISK	5.6.2	
NAME	6.2.3.7	6.3.3
	6.3.8.1	7.2.3.7
DEFINITIONS		
DEBUG REQUEST CARDS	6.3.4	
DEBUGGING PACKAGE	6.3.1	
DEBUGGING PACKET	6.3.3	
EOF CARD	6.2.3.2	7.2.3.2
	8.4.5	
MAGNETIC TAPE	8.4.2	
MATHEMATICAL TECHNIQUES	8.3.1	
PAPER TAPE	9.1.3	
PHYSICAL PROBLEM CODE	8.3.1	
PLOT CARD	8.5.4.1	
STANDARD MONITOR JCB	5.5.0	
DENSITY	4.1.0	8.4.4.2
DUMPS		
DYNAMIC (II)	7.3.0	
DYNAMIC (IV)	6.3.8.2	
POST-MORTEM (II)	6.4.0	7.4.0
POST-MORTEM (IV)	6.3.8.1	6.4.0
DUM03-DUM12	8.4.3	

EAM		
EQUIPMENT	4.4.0	
KEYPUNCH STANDARDS	8.2.2	
USAGE	8.2.1	
EQUIPMENT		
SEE MACHINES		
ERROR		
/.LXERR/	6.3.8.1	
/.LXSTP/	6.3.8.1	

/.LXSTR/	6.3.8.1	
FLOATING PT. OVERFLOW	6.3.9.2	
I/O MESSAGES	6.3.9.1	
TRACE	6.3.9.1	
WALK-BACK	6.3.9.1	
EXAMPLES		
DEBUG REQUESTS	6.3.4	
DYNAMIC DUMP (II)	7.3.6	
DYNAMIC DUMP (IV)	6.3.8.2	
EOF	6.2.3.2	7.2.3.2
JOB MAKEUP (II)	7.2.1	7.2.2
JOB MAKEUP (IV)	6.2.1	6.2.2
PLOT CARD	6.2.3.1	7.2.3.1
	8.5.4.1	
POST-MORTEM DUMP (IV)	6.3.8.1	
SETUP CARDS	6.2.3.4	
FAP	7.6.0	
FLOATING PT. OVERFLOW	6.3.9.2	
FMS	7.0.0	
FORTRAN II	7.2.0	
ASSEMBLER	7.6.0	
CHAIN TAPES	5.7.4	
CONTROL CARDS		
$DECK	7.2.3.11	
$EXECUTE FORTRAN	7.2.3.5	
$JOB	6.2.3.3	7.2.3.3
$SETUP	6.2.3.4	7.2.3.4
*DATA	7.2.3.11	
*LABEL	7.2.3.7	
*SYMBOL TABLE	7.2.3.8	
*XEQ	7.2.3.6	
ASTERISK (*)	7.2.3.10	
EOF	7.2.3.2	8.4.5
CONVERSION FROM	7.5.0	APPENDIX B
DEBUGGING	7.3.0	
G-FORMAT	7.3.5.2	
MONITOR SYSTEM	7.0.0	
FORTRAN IV	6.2.0	
ASSEMBLER	6.7.0	
CONTROL CARDS		
$DATA	6.2.3.11	
$DECK	6.2.3.11	
$IBDBL	6.2.3.6	6.3.3
$IBFTC	6.2.3.10	

$IBJOB	6.2.3.5	
$JOB	6.2.3.3	
$SETUP	6.2.3.4	
EOF	6.2.3.2	8.4.5
CONVERSION TO	7.5.0	APPENDIX B
DEBUGGING	6.3.0	
LOCATIONS	6.5.0	
MEMORY MAP	6.5.0	
SYMBOLS (VARIABLES)		
DEFINED	6.5.0	
VIRTUAL	6.5.0	
IBJOB	6.1.0	
IBSYS	5.3.0	5.3.2
IOCS	5.4.0	
J. O. NUMBER	6.2.3.3	
JOB		
ACCOUNTING	8.3.0 8.3.4	8.3.3
CLASSIFICATION	5.5.0	6.2.3.3
DEFINITION	5.5.0	
DESCRIPTION FORM	8.3.0	8.3.1
HANDLING	8.2.0	8.2.4
OUTPUT	6.5.0	
OVERLAY	6.5.0	6.6.0
PRIORITIES	8.1.0	
STANDARD MONITOR	5.5.0	
SUBMITTAL	6.2.1 8.2.3	7.2.1
TERMINATION	6.2.3.3 8.1.0	6.4.0
JOB CARD	6.2.3.3	
JOB MAKE-UP		
DEBUG (II)	7.2.1	7.2.2
DEBUG (IV)	6.2.1	6.2.2
FORTRAN II	7.2.1	7.2.2
FORTRAN IV	6.2.1	6.2.2
SAMPLE FORTRAN II	7.2.2	
SAMPLE FORTRAN IV	6.2.2	
SIFT	APP. B 2.0	

KEYPUNCHING		
CARD COLOR	8.2.2	
FORMS	8.2.2	
RESTRICTIONS	8.2.2	
SERVICE	8.2.2	
STANDARDS	8.2.2	
KWIC	8.6.0	
LIBRARY		
AMES PROGRAM	8.6.0	
SHARE	8.6.0	
SYSTEM	8.6.0	
USERS MAGNETIC TAPE	8.4.6	
LINES OUTPUT		
COST	8.3.2	
DEBUG MAXIMUM (II)	7.3.5.7	
DEBUG MAXIMUM (IV)	6.2.3.6	6.3.7
ESTIMATED	6.2.3.3	8.1.0
FORMAT	6.5.0	
LINE COUNT	6.5.0	
MAXIMUM NUMBER	6.2.3.3	8.1.0
SYSTEM LIMIT	6.2.3.3	8.1.0
LOCATION, ABSOLUTE	6.5.0	
LOCATION, RELATIVE	6.5.0	
LXCON	6.3.8.1	
MACHINES		
DATAPLOTTERS	4.3.0	8.5.1
	8.5.2	
DCS	4.1.0	5.1.0
EAM	4.4.0	8.2.1
RENTALS	8.3.2	
SPECIAL	4.5.0	
1401	4.5.0	4.6.0
1440	4.2.0	
7040/7094	4.1.0	5.1.0
7740	4.2.0	
MAGNETIC TAPE		
SEE TAPES, MAGNETIC		
MAP	6.7.0	
MATHEMATICAL TECHNIQUES	8.3.1	
MAXIMUM PT. DISPLACEMENT	8.5.4.2	
MEMORY		

ALLOCATION	5.4.0	6.5.0
	6.6.0	
DISK	4.1.0	
LIMITATIONS	5.4.0	6.5.0
	6.6.0	
1401	4.1.0	4.5.0
	4.6.0	
1440	4.2.0	
7040	4.1.0	
7094	4.1.0	
OPTIONS, CONTROL CARD		
DD	6.2.3.10	6.3.2
FILE COUNT	6.2.3.4	
LINE MAX	6.2.3.6	6.3.3
NODECK	6.2.3.10	
NOGO	6.2.3.5	
NOSOURCE	6.2.3.5	
OTHERS	6.2.3.12	
SDD	6.2.3.10	6.3.2
SETUP CARD	6.2.3.4	
TRAP MAX	6.2.3.6	6.3.3
OUTPUT		
ACCOUNTING RECORD	6.5.0	
COMPUTED RESULTS	6.5.0	
DEBUGGING (II)	7.3.0	
DEBUGGING (IV)	6.3.8.2	6.5.0
IBJOB	6.5.0	
MEMORY MAP	6.2.3.5	6.5.0
PRINT TAPE	6.2.3.4	6.5.0
SYMBOL REFERENCE DATA	6.5.0	
OVERLAY	6.6.0	
PAPER TAPE		
SEE TAPES, PAPER		
PHYSICAL PROBLEM CODES	8.3.1	
PLOTTING		
CONTROL CARDS	6.2.1	6.2.3.1
	6.2.3.4	7.2.3.1
	8.5.4.1	
EQUIPMENT	4.3.0	8.5.2
INK COLOR	8.5.4.1	
PLOT CARD	6.2.3.1	7.2.3.1

	8.5.4.1	
PLOT PAGE	8.5.3.1	8.5.4.3
RESTRICTIONS	8.5.4.2	8.5.4.3
ROUTINES	8.5.3.2	
SYMBOLS	4.3.0	8.5.2
USAGE	8.5.0	
POST	6.4.0	7.4.3
POST-MORTEM DUMP (IV)	6.3.8.1	
PRIORITY, JOB	8.1.0	
PRODUCTION DECKS	5.6.7	
PUNCH	6.2.3.4	8.4.5
QUEUES	4.2.0	5.2.0
RECOM	5.5.0	
REDUNDANCY	8.4.2	
REMOTE STATIONS		
1401 ON LINE	4.5.0	
1440	4.2.0	
RENTALS, MACHINE		
SEE COST, MACHINE		
RESTRICTIONS		
ANALOG-DIGITAL CONV.	9.2.0	
DEBUGGING (II)	7.3.5	
DEBUGGING (IV)	6.3.6	6.3.7
ERROR WALK-BACK	6.3.9.1	6.5.0
FORTRAN II CHAIN TAPES	5.7.4	
LINES	6.2.3.3	6.3.7
	7.3.5.7	8.1.0
MAGNETIC TAPE	8.4.3	8.4.4.6
	8.4.5	8.4.7
MAX.NO. OF TAPES	5.7.3	
PAPER TAPE	9.1.2	
PLOTTING	8.5.4.2	8.5.4.3
SENSE SWITCHES	5.7.1	
SIFT	APP.B 3.0	
TAPE LIBRARY	8.4.6	
TIME	6.2.3.3	8.1.0
7094 STOP	5.7.2	
ROOM NUMBERS	APPENDIX A	
ROUTINES		
INPUT-OUTPUT		
DUM03-DUM12	8.4.3	

REASSIGN MODE	8.4.3	
MAGNETIC TAPE	8.4.8	
PAPER TAPE	9.1.6	9.1.7
PLOT	8.5.3.2	
UTILITY	8.4.8	9.1.6
	9.1.7	
ROUTINES, TERMINATION		
.LXCON	6.3.8.1	6.4.0
/.LXERR/	6.3.8.1	
/.LXRTN/	6.3.8.1	
/.LXSTP/	6.3.8.1	
/.LXSTR/	6.3.8.1	
CRASH	6.3.8.1	6.4.0
CRISIS	7.4.2	
ENDFIL	6.4.0	8.4.7
EXIT	6.4.0	
POST	6.4.0	7.4.3
RUN	5.5.0	
SDA	8.6.0	
SDD		
SEE OPTIONS,CONTROL CARD		
SETUP CARD	6.2.3.4	
SIFT	APPENDIX B	
SSD	8.6.0	
STORAGE		
SEE MEMORY		
SYSTEM		
DCS OPERATING	5.1.0	
DEBUGGING (II)	7.3.0	
DEBUGGING (IV)	6.3.0	
FORTRAN MONITOR	7.0.0	
IBJOB MONITOR	5.3.2	6.0.0
RESIDENCE	5.1.0	5.2.0
	5.3.1	
TAPE NUMBER	6.2.3.4	8.4.6
TAPES		
MAX. NO.	5.7.3	
TAPES, MAGNETIC		
BCD	6.2.3.4	8.4.2
	8.4.3	
BINARY	6.2.3.4	8.4.2

	8.4.3	
BLOCKED	6.2.3.4	8.4.2
	8.4.4.5	
DEBLOCKED	6.2.3.4	8.4.2
	8.4.4.4	
DENSITY	4.1.0	8.4.4.2
FILE PROTECTED	6.2.3.4	8.4.2
LOGICAL	6.2.3.4	8.4.3
	8.4.5	
PERMANENT	8.4.6	
PLOT	8.4.4.4	8.5.4.3
PRINT	6.2.3.4	8.4.4.4
	8.4.5	
RECORD SIZE	8.4.3	8.4.4.3
RELEASE OF	8.4.6	
ROUTINES	8.4.8	
UNBLOCKED	6.2.3.4	8.4.2
	8.4.4.4	
TAPES, PAPER		
RESTRICTIONS	9.1.2	9.1.4
ROUTINES	9.1.6	9.1.7
USAGE	9.1.0	9.1.2
TIME		
SEE COMPUTER TIME		
TIMING, HARDWARE		
CARD PUNCH		
1440	4.2.0	
7040, 1401	4.1.0	
CARD READER		
1440	4.2.0	
7040, 1401	4.1.0	
DCS	4.1.0	
MAGNETIC TAPE	4.1.0	
PRINTER		
1440	4.2.0	
7040, 1401	4.1.0	
UPDATE	3.0.0	
WORK ORDERS	8.3.3	